ChatGPT

High-Realism PvE DayZ Server Design (Chernarus+) with AI-Driven Economy & Discord Integration

1. Server Hosting & Configuration

Hardware & Hosting Setup: For initial testing with ~20 players, a hybrid setup is recommended: host the DayZ server on a dedicated OVH **Game** VPS (for strong DDoS protection) and run AI/Discord microservices on a local machine or second server. OVH's "Game" servers include specialized anti-DDoS for gaming [1]. Ensure the server has at least 4 CPU cores @ 3+ GHz (DayZ is heavily single-threaded) and 8–12 GB RAM [2]. An SSD is required for fast read/write of the mission files and persistence data [3]. Start with this modest spec and scale up hardware as player count and mod load increases (DayZ server performance can degrade past ~60 players without high-end CPU).

Network & DDoS: Use OVH's Game server line which **automatically filters DDoS attacks** at the network level [4] [1]. Avoid hosting on basic plans (e.g. Kimsufi/SYS) that lack game traffic mitigation [4]. Keep server ports default (2302-2305 UDP) and behind the OVH firewall. You may consider Cloudflare Spectrum or similar for any web API endpoints if exposed, but game traffic itself will rely on OVH's mitigation. Regularly backup your server's **storage folder** (persistence data) to an off-site location in case of data loss or attacks.

Restart Cycles & Backups: Implement automated restarts every 4-6 hours (with warnings) to **clear memory leaks and reset events**, which is common practice for DayZ stability [5]. Use a restart script that also archives the **storage_1** persistence folder (player progress, bases, vehicles) and logs to a timestamped backup. This ensures you can rollback if a mod update or server crash corrupts data. Plan a maintenance window weekly to update mods and Windows/Linux patches as needed.

Performance Tuning: Limit unnecessary mods and scripts – each mod adds overhead, so include only what's needed for the experience [6]. Monitor server FPS and hitch times via **RCON** or logs, and adjust tick rates if necessary (DayZ server tick is usually fixed, but heavy AI mods may need tweaking spawn counts). In *serverDZ.cfg*, set `simulationCountLimit` and other limits conservatively if using many AI to prevent server overload. Use tools like *Performance Monitor* (if on Windows) or community watchdogs to watch CPU/RAM usage [6]. Keep the server's OS lean – disable unnecessary services, and consider process priority boosts for the DayZ server process.

Core Configuration Files: Configure `serverDZ.cfg` with a descriptive hostname, password (if private during testing), and set `maxPlayers = 20;` initially (can raise later). Enable `verifySignatures = 2;` and `BattlEye = 1;` for security [7] [8]. Set `disablePersonalLight = 1;` to remove the unrealistic ambient player glow at night [9]. Use `disable3rdPerson = 0;` (third-person allowed) or `1` (true first-person hardcore) per your difficulty vision [10]. For a hardcore survival vibe, **enable full night cycles**: e.g. `serverNightTimeAcceleration = 1;` (or low acceleration so nights are long and tense) and perhaps slightly accelerate day to 2x if needed [11]. Keep `serverTimePersistent = 0;` so time resets on restart (or consider a progressive time if wanting changing seasons). Example snippet from config:

```
// serverDZ.cfg important settings
hostname = "MyRealism PvE - Chernarus+";
password = "";  // no password, public
maxPlayers = 20;
disablePersonalLight = 1;
disable3rdPerson = 1;
disableCrosshair = 1;  // no crosshair for realism
disableMap = 0;        // map disabled if you want players to find maps in-game
loginQueueConcurrentPlayers = 5;
loginQueueMaxPlayers = 500;
instanceId = 1;  // ensure unique if multiple servers on same machine
storageAutoFix = 1;    // auto-repair corrupted persistence [12] [13]
guaranteedUpdates = 1;
```

Set up the mission selection to use Chernarus+:

```
class Missions
{
    class DayZ {
        template = "dayzOffline.chernarusplus";
    };
};
```

This loads the vanilla survival mission on Chernarus+ [14], which will be further customized by mods and scripts.

Gameplay Tweaks: Enable the **cfggameplay.json** if custom gameplay settings are needed (stamina, loot multipliers etc.) by adding `enableCfgGameplayFile = 1;` in serverDZ.cfg [15]. In `globals.xml`, you can adjust **environment parameters** like lighting and AI sensory settings. For instance, increase zombie hearing range or adjust night light levels for more tension at night. In `cfgeconomycore.xml`, consider reducing the nominal for food and ammo to enforce scarcity (the dynamic economy microservice will override some values anyway).

Persistence & Map Reset: To achieve *"soft attrition"* of bases and loot over time, configure `lifetimes` in **types.xml** for base building objects and stashes so they eventually despawn if not maintained. For example, set fences and tents to decay after a few weeks unless interacted with (this prevents an infinite buildup of stuff). Use `territories.xml` (from Expansion mod) to define any no-build zones or territory sizes if needed (e.g. prevent building in high-tier areas to keep them contestable). Plan periodic map refresh events (e.g. a storyline "cleanup" or seasonal wipe) where abandoned bases or items are wiped selectively. This could be done via the microservice: it can identify structures not visited in X days and delete them during a scheduled restart.

Server Security & Admin: Use a strong `passwordAdmin` in serverDZ.cfg and tools like **Battleye RCON** with white-listed IP or a VPN for remote admin. Activate `adminLogPlayerHits = 1;` and related logging to catch any suspicious damage or kills – though it's PvE, this helps monitor exploits. Enable the BattlEye filters and use **Community Online Tools** or **VPPAdmin** mod in a limited capacity for easier debugging during setup (remove for live play to avoid temptation of admin abuse – this should be an *"adminless"* server as much as possible). Also configure an automated watch-dog to reboot the server if it crashes (a simple .bat loop or a Linux systemd service with restart always).

Backups: All mission files (xml/json) and mod configs should be under version control (Git) as part of the GitHub SDLC pipeline (see Section 7). Set up a scheduled job to back up the live database (storage folder) nightly. You can use the microservice agent to detect idle times and trigger a backup to cloud storage. This ensures the long-term progression is safe.

2. Mod Selection and Compatibility

We will use a carefully curated set of mods to implement advanced AI, missions, economy, vehicles, and immersion features, while **maintaining realistic survival** (no futuristic or overpowered gear). Below we organize mods by category, with notes on their features, configuration, and compatibility. All mods will be loaded via the `-mod=` or `-serverMod=` launch parameters and their keys added to the keys folder. The load order will generally follow dependencies (e.g. CF, then frameworks, then Expansion, then content mods).

2.1 AI and NPC Enhancements

- **DayZ Expansion AI:** Part of the Expansion suite, this adds **survivor NPCs** that can form squads, patrol, use cover, vault obstacles, and even open doors [16]. Expansion AI provides the baseline for spawning and controlling AI behaviors. When enabled, you can configure patrol routes, spawn intervals, faction allegiances, etc., in the Expansion AI settings (JSON files in `profiles/ExpansionMod/AI/`). We will use Expansion AI to populate the world with roaming survivors and bandits. This mod is server-side but requires clients to have Expansion mod loaded as well.

- **Dynamic AI Spawns (Dolphin's "DayZ-Dynamic-AI-Addon"):** An extension built on Expansion AI framework, it introduces **unpredictable, dynamic NPC encounters** across Chernarus [17]. Players can encounter **hostile bandits, neutral survivors, or even friendly NPCs** anywhere on the map. The spawns are highly configurable (types of AI, their loadouts, difficulty, frequency) via its `SpatialAISettings.json`. This mod makes PvE more lively and dangerous, as no area is truly safe from random encounters. Dynamic AI ensures players must *"stay alert and ready for unexpected encounters"* [18]. We will tune it so that coastal areas might have occasional friendly survivor NPCs (who might trade or give hints), whereas military zones have heavily armed hostile factions. *Compatibility:* It requires DayZ Expansion Core and AI; it runs server-side only (list in `-serverMod`), so players don't need extra downloads beyond Expansion. We'll ensure to get **Expansion-Spawn-Selection** mod as recommended, to prevent AI spawning while players are selecting spawns [19].

- **"V++ AI" / VanillaPlusPlus AI:** (If available – this may refer to an old project or custom AI) We will investigate if any **Vanilla++ AI** mods exist that offer alternate AI systems. However, with Expansion AI and Dynamic AI in place, we likely have the best-supported solution. We'll skip redundant AI systems to avoid conflict. (Possibly the user meant **VPPAdmin Tools** integration for spawning AI, but that's not needed with the above).

- **Infected (Zombie) Tweaks:** To increase the PvE challenge, we won't rely solely on mods – we will configure **events.xml** to increase infected counts in key areas and use mods for their behavior:

- Increase zombie counts by raising the `<nominal>` values in **events.xml** for infected events (e.g. "CityInfected", "VillageInfected") and adjust `Max` in **globals.xml** to allow more total zombies. Also use **cfgenvironment.xml** to make infected more aggressive if possible.
- Use a mod like **Zombies Health Rebalance** to make the undead tougher (e.g. requiring headshots). For instance, *Zombies-Health-Rebalance* mod can adjust health so common guns no longer one-shot every infected [20]. Alternatively, use **WalkingDeadZombies** mod to make them slower but much more numerous/hard to kill [21]. We will likely prefer faster, dangerous zombies (to keep players on edge), so rather than slow them, we increase their damage and resilience slightly.

- **Infected Behavior Mods:** Consider **Morty's Infected** or any AI enhancement that allows infected to roam between towns or migrate, making the world feel dynamic. If none available, the Dynamic AI mod may suffice by also spawning animals or creatures (if supported).

- **"Raiding AI"** (conceptual): We intend to configure some AI squads that **attack player bases or key points**, to simulate world threats. While not a specific Steam mod by that name, we can script this via Expansion AI or the mission system – e.g. a bandit raid event that targets a random base at interval. This will be handled in the Mission system (see 3. PvE Missions), spawning AI raiders via server events. We'll ensure such raiders are balanced (they won't have explosives to offline raid everything, just small arms to harass bases). The idea is to keep bases from being 100% safe even in PvE – environmental threats exist.

- **NPC Quest Givers & Friendly AI:** We will use Expansion's **Static AI spawns** at safe zones (trader hubs, see below) to act as quest givers or guards. These AI will not attack players (faction set to friendly) and can provide dialogue or at least atmosphere. If possible, we'll integrate **Expansion AI's voice** (DayZ Expansion has an Audio module that can give AI spoken lines [22]) for immersive interactions, though custom voice content may be needed.

Configuration & Conflicts: Running multiple AI mods means careful balancing. Expansion AI with Dynamic AI is designed to work together (Dynamic AI actually **requires Expansion AI framework** [17]). We will avoid older mods like SurvivorAI or others that conflict with Expansion. The server's CPU load will increase with AI – we'll cap simultaneous AI spawns and use Expansion's group system so not every player has their own horde. All AI mods will be in `@ExpansionModAI`, etc., and any settings JSON (e.g. *spatialsettings.json* from Dynamic AI) will be edited to ensure no *alien or monster* AI – only human NPCs and possibly wolves/bears (vanilla animals).

2.2 Mission & Quest Frameworks

- **DayZ Expansion Quests:** This will be our primary **quest/mission framework** for structured, story-driven content. The Expansion Quests mod allows creating MMO-like quests with varied objectives (actions, item collection, delivery, elimination, travel, treasure hunts) [23]. We can define quest chains that unlock sequentially, fulfilling the "progression locked areas" concept. Expansion Quests supports multi-objective quests and even group quests (when combined with Expansion Groups) [24]. We'll leverage this to create a **main storyline** across Chernarus:

- Example: a series of quests where survivors must **restore a radio transmitter**, then **secure a settlement**, then **explore a contaminated military bunker**. Each quest completion triggers the next and possibly "unlocks" a new area (e.g. after restoring power, the doors to an old bunker can now be opened with a keycard).
- The quest types cover our needs: *AI quests* (clear an area of AI or escort an NPC) are possible when combined with Expansion AI [25] – we will use *AI Camp* objectives to have quests like "Clear the bandit camp in the Black Forest" or *AI Escort* to escort a doctor NPC to a field hospital [25]. Delivery quests can simulate supply runs (e.g. bring blood bags to a medical unit).
- Quest configuration is done via JSON files in `ExpansionMod/Quests/`. We'll define quest NPCs (with `QuestNPCs.json`) who give out missions at hubs, and quest definitions in `Quests.json`. Because the Expansion framework is robust, we'll prefer it for persistent, character-bound quests (it can even track quest completion per player and we can reset progress as needed via commands [26]).

- **Survivor Missions (server-side mod by [Funkdoc])**: In addition to structured quests, we introduce *Survivor Missions* to provide **dynamic, repeatable PvE events** that anyone can undertake. This mod runs entirely on the server (no download for clients) and broadcasts missions over the in-game radio channel [27]. Players who find a radio and tune to a certain frequency will receive missions like: *"An airdrop of medical supplies has landed near Vybor, retrieve it"*, *"Clear infected at the Green Mountain summer camp"*, *"Rescue a stranded survivor (NPC) in Berezino"*, or even quirky ones like *"Free the pigs from a farm"* [28]. There are 1400+ mission locations predefined across Chernarus, Livonia, Deer Isle etc [29], ensuring variety. **Rewards** are given for completion (could be gear, vehicles, reputation, or currency).

- Survivor Missions mod basically creates a **mission zone that attracts PvE and possibly PvP** [27]. Though our server is PvE, it still creates a risk/reward as multiple players might rush to complete it cooperatively (or competitively for rewards).
- We will customize or write new missions via its scripting API (the mod is well-documented for adding custom missions [30]). For example, to fit our server lore, we might add missions that tie into factions: a radio call from "Freedom Fighters faction" asking players to eliminate a rampaging infected horde in a town, etc.
- *Integration:* Because Survivor Missions operates via radio and server messages, it should not conflict with Expansion Quests (which are more personal and NPC-driven). They complement each other: Expansion Quests = **story progression and faction quests**; Survivor Missions = **emergent dynamic events**.

- **Additional Mission Mods:** We will avoid duplicating frameworks, but note that **"DZ Missions"** (if referring to community missions packs) or **"VPP Missions"** might be alternate names. One known community mod is *"SM_VPPM"* which adds mission markers on the VPP Map for Survivor Missions [31]. If we use a custom map UI (say, VPP Admin's map for players or Expansion's map markers), we might integrate mission markers. In Expansion, we can also use the **Expansion Navigation** mod to place markers for quest objectives [32] (so players see waypoints for active quests, maintaining accessibility).

- **Keycard & Locked Areas:** To implement *progression-locked areas*, we will use the **KeyCard Rooms mod** (by HellCat) [33]. This mod provides mappable "locked rooms" or bunkers that require a specific keycard item to enter. For example, we will place a **hidden bunker in Tisy or Novaya** – initially inaccessible. As part of a quest line, players will obtain a *"Key Card A"* which grants access. The KeyCard mod allows multiple colored cards and corresponding doors [34]. We'll configure a high-tier loot bunker that effectively serves as endgame content once unlocked.

- We'll tie this to a quest: e.g. Quest: *"The Cure": Find the access card to the secret underground lab* – which upon turning in, triggers an event unlocking the bunker (by spawning an unlocked door or just informing the player they can now enter).
- *Note:* Livonia's 1.19 update introduced an underground bunker with a punchcard [35] – since we are on Chernarus, using the mod and/or custom JSON (like scalespeeder's Novo bunker json [36]) will achieve a similar effect on our map.

- **Mission Difficulty & Scaling:** Both Expansion Quests and Survivor Missions allow adjusting difficulty. We will script **mission progression**: early missions (Tier 1) will involve simpler tasks (hunting, scavenging, fighting a few infected). Later missions (Tier 3+) might require coordinated groups – e.g. clearing a large area of heavily armed AI. We can lock these behind earlier quests completion, ensuring a *difficulty curve* that matches player skill and gear progression. The microservices (section 6) can also dynamically tune mission difficulty: for instance, if the server detects all players are well-armed, it can inject more AI into missions or tougher enemy types (like more armored zombies or higher-tier bandits).

- **Faction Missions & Reputation:** We will implement a basic **faction reputation system** (likely through our custom microservice, but integrated with quests). Factions are described in detail in Section 3; here's how missions tie in:

- Certain quests will be offered by specific factions (e.g. a trader's guild might ask you to deliver goods, a military remnant might ask you to gather intel). Completing these yields reputation points.
- We can use Expansion Quest's ability to run **custom script on completion** (if available) or simply have our microservice listen for quest completion events via log and update a rep database. High rep might unlock special quests or trader discounts.
- Faction-based mission example: *Medical Team* – a roaming "Medics Without Borders" faction posts tasks to rescue injured NPCs or deliver medical supplies, improving your standing with them (leading to rewards like better medical items or a paramedic vehicle spawn).

Compatibility: Expansion Quests requires Expansion Core, CF, and Dabs Framework (we will load all required mods in proper order) [37]. Survivor Missions being server-side means no client conflict, but we must run it as `-serverMod=@Survivor Missions` and ensure the launch parameter is supported by our host (some GSPs require tricks to load server-only mods [38]). We will test that thoroughly. Neither mod introduces major conflicts with others; they mostly operate via their own scripts and data files. Just be mindful to not use another quests mod in parallel (to avoid player confusion).

2.3 Trader, Economy & Market Mods

- **DayZ Expansion Market (Trader):** We will utilize the Expansion's built-in trader system to create our **trading economy**. Expansion Market comes with a UI that allows buying/selling items at configured NPC traders. It supports categories, price controls, and is tightly integrated with Expansion (including safe zone territories, etc.) [39] [40]. Key features:

- We will set up multiple trader NPCs (using Expansion's Trader NPC entities) in safe settlements (e.g. Green Mountain, Kumyrna, or a custom trading post). These traders will have **limited stock** to keep economy in check. Expansion Market config (in `profiles/ExpansionMod/Market`) allows setting stock quantity and restock timers.
- **Black Market Trader:** We plan a hidden Black Market that sells high-end gear (rare weapons, suppressors, explosives) at steep prices and possibly only at night or via special access. If Expansion Market doesn't natively support a "flag" for black market, we simulate it by placing that trader in a remote, dangerous area (e.g. an island or radiation zone) so players must risk reaching it. Alternatively, we integrate a mod specifically for Black Market trading zones, such as *The Architect's Black Market* custom building mod [41] which provides a special trader base design. In any case, the **Black Market** will use the same Expansion trader mechanics but with separate config files (we'll have a `BlackMarket.json` category list). It will only buy/sell contraband (e.g. drugs, rare ammo, Tier 4 guns).

- Expansion Market supports **dynamic item pricing** indirectly through stock levels (if we configure the `inflation` parameter): as stock dwindles, prices can rise, etc. However, for full dynamic economy we will mostly rely on our economy microservice to adjust prices between restarts (see section 4 and 6).

- **TraderPlus (alternative economy mod):** If we find Expansion Market lacking a feature (like multi-currency or banking), we could consider TraderPlus mod. TraderPlus offers dynamic pricing based on stock and even multiple currency support [42] [43]. It also includes a player banking system and licenses (to gate high-tier purchases behind a "license" item) [42]. This is attractive for realism (e.g. requiring a pilot license to buy a helicopter). **However**, TraderPlus is a separate system and might conflict with Expansion's traders. We likely will **not use TraderPlus simultaneously** to avoid duplicate systems. Instead, we'll mimic those features: for currency, Expansion uses a currency item (we can designate rubles, dollars, or bottle caps as currency – possibly using a modded item if needed). For licenses, we can simply treat them as key items (players must show an item or complete a quest to unlock certain trader menus).

- If we did switch entirely to TraderPlus, we would get dynamic pricing out-of-the-box: "Stock levels influence prices – more stock means lower prices, adding strategy to trade" [44]. TraderPlus also has an integrated car lock system and safe zones. But given we already use Expansion heavily, it's simpler to stick to Expansion Market for consistency.

- **Economy Configuration (types.xml & Trader Config):** We will heavily customize **types.xml** to set item nominal values in line with our economy:

- **Scarcity:** Reduce spawn rates of military weapons, NVGs, high-caliber ammo. Only basic civilian weapons and supplies should be relatively findable. This makes the trader economy meaningful (players can't just loot everything easily).
- **Tiered loot:** Chernarus is divided into loot tiers by location (vanilla). We'll keep that, but our mission system might introduce rare items as rewards that don't spawn otherwise.
- The **trader price list** (Expansion uses JSON categories with each item's buy/sell price and quantity). We will create this likely in a spreadsheet or Baserow, then export JSON. Prices will follow a realistic logic: common food and tools cheap, ammunition moderately expensive, medical supplies valuable (to push players to consider crafting or missions for them), and top weapons extremely costly (and

possibly only in black market). We'll also incorporate **bartering**: e.g. the Black Market might not take cash for certain items but require trade-in of specific loot (e.g. exchange 5 rare artifacts or data drives for a suppressor).

- **Multiple currency**: If using Expansion, usually one currency item is defined (like stack of rubles or gold nuggets). Instead, we could designate, for realism, a **two-currency system**: e.g. use **coins** for general goods and **networked value (like USB sticks)** for high-tech (in lieu of dollars). This is complex to implement directly in Expansion, so more likely we stick to one currency but have some items only obtainable via quests (not purchasable at all).

- **Modded Items in Economy:** Mods like **SNAFU Weapons** (see 2.4 below) add a huge variety of guns. We will integrate them carefully: not all mod weapons will appear in traders. Perhaps basic hunting rifles from SNAFU can be in trader, but military-grade assault rifles might remain loot-only or quest rewards. This prevents trivializing end-game gear. We will use a community-made Expansion Market config for SNAFU as a base (there are Expansion trader config files for SNAFU on GitHub [45]).

- **Safe Zones & Trader Hubs:** Use Expansion's **SafeZone** module or territory flags to designate the trader areas as no-PvP (and perhaps no damage at all) and no AI spawn. Since it's PvE, players won't be killing each other anyway, but safe zones ensure no stray zombies or environmental harm. We might script that if a Dynamic AI trigger tries to spawn bandits near a safe trader, the AI either despawn or become neutral in that radius.

Compatibility & Config Paths:
Expansion Market config files: located under your server profile in `ExpansionMod/Market/*.json`. We'll have files like `TraderCategories.json`, `MarketItems.json`, etc. Ensure any mod items (SNAFU guns, new vehicles) are added to these with proper class names and prices. We have to also include the mod's types.xml entries to spawn those items in the world (if we want them lootable). The **Trader mod (Dr. Jones)** is not used in parallel with Expansion to avoid conflicts; if we were, it stores config in a `Trader` folder with files like TraderObjects.txt etc., but we'll not use that. TraderPlus stores config in the mission folder under `TraderPlus` if it were used.

We will validate our economy XML/JSON with tools (expansion provides a validator, and community tools like DZconfig). Also, keep an eye on any mod that introduces its own economy config (e.g. if a mod has an `cfgspawnabletypes.xml` override for loot distribution – we'll merge those).

2.4 Vehicles and Helicopters

- **DayZ Expansion Vehicles:** The Expansion mod brings a host of new vehicles: cars, trucks, boats, and notably **helicopters** [46] . Out-of-the-box, Expansion adds vehicles like the MH-6 Little Bird helicopter, gyrocopters, and others, plus improved vehicle physics. We will enable Expansion Vehicles module to have a foundation of relatively **realistic vehicles** (no crazy supercars; mostly re-textured ARMA vehicles or community models that fit the apocalypse). Key vehicles we plan:
- **Ground Vehicles:** 4x4 offroad, a military UAZ or Humvee, perhaps a bus for group transport, and some civilian cars. No high-end sports cars; focus on practical vehicles that suit Chernarus terrain. We might include a mod like *MuchCarKey* or similar for vehicle keys if not using TraderPlus's car lock, to prevent anyone from stealing cars in safe zones without the key.

- **Helicopters:** Expansion provides variants like the Merlin, UH-1H Huey, and small helis [47]. We will include at least one small scout helicopter (e.g. Little Bird) and one medium transport (Huey). All helicopters will be *rare*: they won't spawn on every restart. We set up **helicopter crash events** (via events.xml or Expansion events) but for obtaining a *working* helicopter, players likely need to either:
    - Purchase from a trader (very expensive, and possibly requiring a "pilot license" quest).
    - Or find and repair one that spawns damaged on the map (we can use a Vehicle Spawner mod or custom events to have one heli spawn hidden, needing parts).

- **Boats:** If map usage of sea is desired, Expansion also has boats. Chernarus has a coast, so maybe a small fishing boat or dinghy could be included, enabling sea fishing or reaching an island (like Prison island or any custom island we add).

- **RedFalcon Flight System Helis:** For a more **advanced, realistic helicopter flight** experience, we will integrate the **RedFalcon Flight System: Heliz** mod. This mod is a standalone helicopter system that includes many real-world helicopters with detailed physics (including things like autorotation, instrument panels, etc.) [48]. RedFalcon's helis (e.g. MH-6 Little Bird, Bo105, Bell 429, Blackhawk, Chinook, etc.) offer a far more **immersive piloting experience** than Expansion's simpler flight model. We plan to include a limited selection from RedFalcon:

- The **MH-6 Little Bird** (light chopper) for recon and quick travel.
- The **UH-1H "Huey"** for a classic survivor group transport.
- Possibly the **Robinson R22** trainer helo with nerfed controls [49] – this is great for events/training players in flying.
- Avoid armed gunships like the Apache by default (if included, we will disable its weapons or not provide ammo, to keep balance – no flying tanks).
- RedFalcon Helis require some integration: they come with their own JSON config for flight settings and spawn config [50]. We will ensure to place their spawn points via the mod's instructions or allow them to be purchased at the black market.

- *Compatibility:* Running Expansion and RedFalcon together is generally fine (they operate independently). We just need to ensure we don't duplicate identical models (if both mods have a Huey, perhaps choose one). RedFalcon's helis might be considered superior, so we might favor them and not spawn Expansion's own helis. On the client, players will need both mods, but that's acceptable. We'll make sure to load RedFalcon *after* Expansion in mod order so that if any configs overlap, RedFalcon's take priority for its helis. RedFalcon's mod is frequently updated (tested with DayZ 1.28 as of Aug 2025) [51], so we keep it updated.

- **Vehicle Spawn & Persistence:** We will likely not let vehicles litter the map in unlimited numbers. Using **cfgeventspawns.xml** or Expansion's VehicleSpawner settings, we set a cap: e.g. only 10 functional vehicles spawn naturally. Additional vehicles must be earned (bought or mission reward). Helicopters especially: perhaps only 1 or 2 on the server at a time. If a heli is destroyed, an event (maybe an AI helicopter crash mission) could later introduce a new one. We'll also schedule vehicle wipes or maintenance to remove abandoned wrecks to save performance.

- **Fuel & Maintenance:** Increase realism by making fuel scarce. We can use the Expansion Fuel system where players have to maintain generators at gas stations or find fuel cans. Possibly integrate a **gasoline mod** that makes fuel finite at pumps. Repairs will require specific parts (engine parts, rotors, etc.), which we control via loot tables and trader. We can even incorporate the **vehicle parts**

wear feature (some mods or Expansion allow engines to degrade). No indestructible vehicles – even bases can't perfectly protect them (but safe zones will be no-PvP so players can't steal or destroy each others' in safe hub).

- **No "Alien" Vehicles:** We will avoid any mods that add out-of-lore vehicles (no UFOs, no futuristic cars). Everything should be mid-1990s tech at latest, fitting the DayZ environment. For example, **no armored tanks** or APCs. Even if RedFalcon offers a military Chinook, that's fine because it's a real helicopter, but something like an Osprey (which RedFalcon plans) we might not enable unless as a special event.

Config Files: - Expansion vehicles are configured in `cfgspawnabletypes.xml` (for attachments like making sure cars spawn with certain parts) and `VehicleSpawnZones` if using Expansion's spawning. We'll edit those to include new vehicles from mods. - RedFalcon uses its own config file (they provided a PDF guide [52]). We'll follow that to adjust any settings (like make sure advanced flight is on, or adjust any control difficulty if needed to not be too frustrating). - Also, map coordinates for any static vehicle spawns we script (we can use Editor to place them and export to init.c or events). - For crash events: use **events.xml** to create custom dynamic events for downed helicopters (could spawn wrecks with loot and maybe AI around). We might incorporate one mod like *Aircrash* that provides enhanced heli crash sites, or do it ourselves by adding more HeliCrash event entries with mod loot (like maybe SNAFU weapons only appear at crashsites).

2.5 Base Building and Storage

- **BaseBuildingPlus (BBP):** This mod greatly expands base construction options with more buildable objects, snapping mechanics for easier building, and upgradeable parts (e.g. wood to metal walls) [53]. BBP also introduces destructible locks, gates, etc., ideal for a long-term server. Features:
- **Snapping & Advanced Structures:** Players can precisely align walls, floors, stairs, making much more aesthetic and functional bases [53]. They can build larger compounds, watchtowers, and use features like cinder walls, concrete, bunkers if enabled.
- **Raiding Mechanics:** BBP includes features like breaching charge support (if using a raiding mod) and generally balanced health for walls to require significant effort to break [54]. On our PvE server, "raiding" would only be by AI or as decay, but we might still allow base damage via explosives for events (e.g. if a horde event targets a base).
- **Storage**: BBP often pairs with mods like **CodeLock** and **BuildAnywhere**. We will include **Code Lock** mod to allow players to put 4-digit combination locks on gates and doors (instead of vanilla keys which are limited). CodeLock is widely used and should integrate with BBP gates easily.
- **BuildAnywhere:** Likely include this so players can place objects without as many restrictions (preventing frustration when a flat ground is hard to find). Expansion also has a "Build Anywhere" setting; we will ensure one consistent method is used (maybe just rely on Expansion's if present, or use the mod).

- We'll need to test compatibility: Expansion BaseBuilding and BaseBuildingPlus can conflict because both modify base building kits and recipes. We will **disable Expansion's base building** (Expansion has config to turn off certain features). This way, BBP is the sole building system aside from vanilla. Expansion's territory system can still be used for claiming land if desired, without using its build parts.

- **DayZ Expansion Base Building:** Alternatively, we could rely on Expansion's base building module. It adds some new parts (like concrete walls, safes, codelocks built-in, etc.). It's more limited than BBP's offerings though. Since BBP is popular and offers more, we chose BBP. But we will use **Expansion Personal Storage** mod for player physical storage like lockers (if quest integration needed) and possibly **Expansion Territory** to allow group base ownership. (Territory flags from Expansion let a group register a flag pole and get build rights – we should see if that works with BBP parts; likely yes, as BBP doesn't have a territory concept, we can simply not restrict building by territory for PvE, or use it if we want to confine bases away from high-loot zones).

- **Storage Mods:** Apart from BBP (which itself adds containers), we'll use some quality-of-life mods:

- **MuchStuffPack** or **Mass's Many Item Pack (MMIO)** for extra furniture and containers (crates, cabinets) to decorate bases.
- **Vaults & Safes:** Possibly include *Schana's Mod* or something for safes that require codes. Expansion has a Safe item if Personal Storage mod is used.

- **Unlimited Stamina for Building:** Perhaps give a bit of help by temporarily boosting stamina when building via a gameplay setting, since carrying logs and metal can be tedious – or encourage use of vehicles for transport instead (more realistic).

- **Base Decay:** As mentioned, we set lifetime in types.xml for base parts. If using BBP, follow its guidance for `BBPSettings.json` (it has a config for decay, IIRC). We might script our microservice to periodically flag structures that haven't been used (e.g. if no one in that base's territory in 2 weeks, mark it for cleanup). This prevents ghost towns of unused bases.

- **No Overpowered Structures:** We will **not** allow indestructible objects. Even metal walls will have a health value. No "safe" that cannot be broken – everything should yield to sufficient time or resources (for PvE, that might mean an admin event or just decay). We won't include any glitchy structures either (like terrain exploits).

Compatibility: Ensure to load **Community Framework (CF)** first (required by many mods like CodeLock, BBP). Then CodeLock, then BBP. Expansion mods usually come after CF too. CodeLock works fine with BBP gates as long as the classnames are configured in CodeLock's compatibility (we'll verify if needed). We should avoid multiple mods doing the same thing (e.g. don't use two different codelock mods). BBP's storage might conflict with Expansion's if both adding same container type (small chance). We'll test base building thoroughly in a closed environment before going live.

The **config files** for these: - BBP has a `BaseBuildingPlus.json` (or similar) in profile for settings like damage, placement, etc. - CodeLock might have no config (just works, or a JSON for which doors can use it). - We also adjust types.xml to add BBP items (nails, etc.) spawn rates and trader configs to sell things like nails, metal wire, combination locks (but at a cost to keep base building as an earned privilege).

2.6 Survival & Immersion Mods

To enhance the hardcore survival experience, we'll add mods that deepen survival mechanics:

- **Terje Skills (Skill System):** A comprehensive skill system mod that brings an **RPG-like progression** to DayZ [55]. TerjeSkills adds 9 skills (Immunity, Medicine, Metabolism, Athletics, Strength, Stealth, Survival, Hunting, Fishing) with 100+ perks total [56] [57]. This directly aligns with our server vision: players improve at tasks through practice – e.g. as you fish more, you catch fish faster; as you do medical actions, you become a better healer, etc. Importantly, skills are **persistent across deaths by default** (though can be configured to drop on death partially) [58], meaning long-term characters have value, encouraging players to stay alive. This mod will give veteran players an edge in late-game areas, which is good because missions will ramp up in difficulty. It requires **TerjeCore and TerjeMedicine** mods as well (TerjeMedicine presumably adds expanded medical illnesses to make the Medicine skill meaningful). We will include those. With TerjeSkills, features like *"hardcore survival with skills, medicine, hunting, fishing"* are naturally implemented:
- **Medicine Skill & TerjeMedicine:** More complex diseases and treatments. Players with higher Medicine skill can treat wounds or sickness more effectively (perhaps craft more potent remedies or get better results from bandages) [59].
- **Immunity Skill:** Characters can fight off infections better at higher levels, which is critical when we have more zombies and perhaps modded diseases (like cholera outbreaks).
- **Hunting & Fishing Skills:** Increases yields from animals, ability to set traps, etc [60]. This synergizes with our plan to emphasize living off the land (especially if loot is scarce, hunting/fishing becomes important).
- **Stealth Skill:** Helps avoid detection by AI (infected or NPC) [61]. This is great for solo players who might want to avoid fights.

- We will configure **Skills.cfg** to maybe make death penalize some XP (so dying is costly, but not a full reset) [58]. This maintains the hardcore feel.

- **Advanced Medical Mods:** In addition to TerjeMedicine (which likely adds new medical items and injury mechanics), we could incorporate **Medical Attention (Updated)** mod, which was known to add realistic medical conditions: fractures that require splints, surgical kits, blood transfusion improvements, etc. If TerjeMedicine covers this, we might not need MedicalAttention separately. But if needed:

- *MedicalAttentionUpdated* mod provides things like detailed wound treatment, blood type importance, and a requirement to treat fractures realistically (not just a splint and instant fix) [62]. We will only add one medical overhaul to avoid redundancy. We suspect TerjeMedicine may already handle these since Terje has Immunity and Medicine skills that presumably tie into such mechanics.

- We will test the medical gameplay: e.g. getting shot might require removing bullets (some mods add that), or severe bleeding needing IV fluids. The goal is to make having a **medic teammate very valuable**. This addresses the "medic teams" point – players might choose to specialize in Medicine skill and effectively become the group doctor, since their higher skill yields better outcomes (just like class roles in an MMO).

- **Nutrition and Diseases:** To further hardcore survival, use **Expansion's Nutrition system** (if available) or mod like *MoreMedicaments*. For example, require vitamins to maintain immunity, track calories and water more rigorously (we could adjust hunger/thirst rates in cfggameplay.json for higher consumption so players must fish and hunt more often). Possibly integrate **Z Virus mod** (if exists) to simulate the virus infection that players must manage (but careful not to annoy players too much).

- **Environmental Threats:** Enable **Expansion's Toxic Zones** (ContaminatedAreas) to create hazardous areas that require protective gear (gas masks, NBC suits). For instance, Tisy military zone could be a static toxic zone until a quest is done to "clean it" (or permanently, requiring gear). This adds a layer of endgame challenge and makes finding an NBC suit part of progression. We just have to place gas zones via cfgEffectArea.json (one in a high-tier loot spot, maybe one random moving zone to keep players on toes).

- **Ambient Lore and Collectibles:** We'll add the **SF Radio mod (Cassette Player)** [63] and possibly **Zen's Music Mod** to have **cassette tapes** in the loot. This allows players to find and play music or recorded messages in-game, enhancing immersion:

- SF Radio mod adds physical radios and cassette items that can play custom audio files [63]. We will curate some in-world "lore tapes" – e.g. a diary entry or military broadcast recorded on tape, that players can listen to for story clues. These tapes can hint at missions or just build the world narrative (e.g. a survivor's last words explaining the origin of a faction).
- Also it allows us to incorporate entertainment – survivors might collect music tapes of various genres (fits the 90s era vibe). It's an optional fun activity that adds depth without breaking realism (cassette tech is old but fitting).

- Config: we'll create custom audio files and instruct players to download them as part of the mod (since SF Radio allows server owners to add tracks [64]). No DMCA songs, we'll use either royalty-free or original voice-acted content for story tapes.

- **Map & POI Enhancements:** We keep the map as Chernarus+, but will sprinkle some **new Points of Interest** via custom mapping:

- Use **DayZ Editor** to create a **race track at Novy Lug** (or use the mod provided) because the user wanted "a single race track." The *Novy Lug Race Track* mod adds a full race complex with large and small tracks, pit area, etc [65]. We will incorporate this mod, which comes as an Editor file or Expansion map file [66]. The race track will be a place for events (player-organized races or time trials). It's a non-traditional DayZ activity, but in a long-term PvE server, these fun diversions are great.
    - We'll set up a **Vehicle Time Trial mission** at the track (maybe Survivor Missions can support a mission like "complete a lap in under 3 minutes" with a reward). Or schedule community race events via Discord.
    - The mod requires EditorLoader or can be directly integrated with Expansion's object spawner [67]. We'll likely use Expansion's method for simplicity (copying the `RaceTrack.map` to the server's expansion objects folder [68] and adding the mapgrouppos entries so loot spawns in the new buildings [69]).
- Other custom POIs: Possibly add a few small camps or hideouts relevant to factions (e.g. a hidden **black market outpost** in a forest, a **scientist's underground lab** entrance using objects, or a **faction base** like a fortified compound for a friendly faction). These can be done via Editor and

placed as JSON. We must be careful not to break DayZ's AI navmesh drastically. But since Expansion has a built-in object spawner, we'll lean on that.

- **Prison Island expansion**: maybe place a trader or mission target there.
- **Bridges**: If we want to ease map traversal, add a bridge to Skalisty Island or from the mainland to Prison Island – but only if it fits realism (a war-time makeshift bridge, etc.). There are mods for bridges if needed.
- **Sound and Atmosphere:** Use mods like **Immersive Soundscapes** to enhance environment audio (forests, night sounds, etc.). Also consider **Infected Audio** mods to make zombie sounds scarier. **Lighting mods** or enabling darker nights (we already will do via config) plus encouraging player use of flashlights (which in dark nights, with no personal light, will be essential – leading to tense moments).
- **Quality of Life Tweaks:** Only those that don't break realism. E.g. **Ear Plugs mod** (to reduce ambient noise when needed), or **Better Hands** (so putting away items is easier). No GPS or AI scan tools. Possibly a **Map mod** that allows marking positions if the player has a physical map & compass (some servers allow seeing self on map only if you have both items – we can script that via a client mod, or just trust players to not use the map if we want hardcore navigation).

All these survival mods combined create a **hardcore survival** environment where players must manage health, learn skills over time, rely on fishing/hunting for food (since canned goods are rare), and benefit from teamwork (e.g. someone with high hunting skill provides food, someone with high medical skill keeps everyone alive).

Compatibility: Many of these immersion mods require **Community Framework** (we have it). TerjeSkills is a newer mod (Nov 2024) and should be compatible with Expansion; we must ensure any overlap with Expansion's (Expansion also had a basic party system and maybe a basic stamina skill, but not as extensive – should be fine). We will place TerjeSkills late in load order (after other mods) to override where needed. Also check that TerjeMedicine doesn't conflict with MedicalAttention if we had both – likely we won't use two medical overhauls; we stick to the Terje suite.

Configuring these mods: - TerjeSkills: config in profile `TerjeSettings/Skills.cfg` (we'll adjust death penalty and any XP rates). - Expansion effects (toxic zones) in `cfgEffectArea.json`. - SF Radio mod: ensure its classnames are added to types.xml to spawn cassettes and radios (the mod provides a types snippet). - Race Track: follow install instructions to spawn it (no .pbo to load, just objects via Editor or Expansion). - CodeLock: usually config via `CodeLock.json` (to specify which doors allowed, etc.).

2.7 Mod Installation & Load Order Summary

Load Order (Suggested):
1. **Community Framework (CF)** – always first.
2. **Community Online Tools or Dabs Framework** (if needed by Expansion Quests [37], Dabs Framework likely auto-included).
3. **DayZ Expansion Core** (and Expansion Licensed, Expansion Assets if applicable – the Bundle mod or separate items) [70].
4. **Expansion Modules**: Expansion Vehicles, Expansion AI, Expansion Market, Expansion Quests, Expansion

Groups, Expansion Personal Storage, Expansion Navigation, Expansion MapAssets (load all that we use).
5. **BuilderItems** (for custom map objects if any placed, also required by race track mod) [71] .
6. **BaseBuildingPlus**, **CodeLock** (BBP might come with BuildAnywhere or we include BuildAnywhere separately).
7. **SNAFU Weapons** (and any other weapon/gear mods like Mass's, etc., to ensure their items load for traders).
8. **RedFalcon Flight System Heliz** (and its Core if separate).
9. **Survivor Missions** (as -serverMod, doesn't affect client load order).
10. **TerjeCore, TerjeMedicine, TerjeSkills** mods.
11. **MedicalAttention** (if used, but likely just Terje covers it).
12. **SF Radio (Cassette)** mod.
13. **Any other smaller mods**: (Map, Sound mods, etc.) – these generally can go last.

We will carefully check each mod's requirements on its Workshop page (many list required items which Steam will auto-subscribe). For example, TerjeSkills requires TerjeCore [72] ; Expansion Quests requires Dabs Framework and CF [37] – we covered those.

Overrides & Conflicts:
- If two mods alter the same item (e.g. both SNAFU and another mod add an M4 variant with same class), class name collisions could occur. We'll avoid adding overlapping weapons packs to prevent this. SNAFU itself has many guns and notes that it's tested with other mods but compatibility not guaranteed [73] . We'll keep an eye out for any RPT errors indicating class conflicts and remove or adjust as needed. - SNAFU and Morty's Weapons together might be overkill. We might stick to one big weapons pack (SNAFU) for consistency, and possibly include a smaller melee pack for variety. - Expansion plus BaseBuildingPlus: disable Expansion build via mission init or config to let BBP handle base parts. - Expansion plus RedFalcon helis: in config, decide whether to disable Expansion's flight auto-hover or any conflicting key binds. Both can exist; players will learn RedFalcon helis handle differently. Document it in server info.

Finally, we will maintain a **mod list documentation** for players, explaining which mods are used and any client-side considerations. All mod licenses (e.g. SNAFU's no reupload policy [74]) will be respected – we will not repack any mods, just use as is and require players to download from Steam (or our server launcher).

3. PvE Factions, Quests and Mission Design

One key to a living PvE world is the presence of **factions** – groups with distinct identities, goals, and relations to the player. We will base faction themes on the assets provided by mods (pre-built outfits, items, and NPC types) as much as possible, to minimize custom model work. Below we propose several factions and how they manifest in game:

- **Survivor Union (Civilian Faction):** A loose coalition of survivors trying to rebuild society. They operate the main **Trader Hubs** (e.g. at Green Mountain or a repurposed town like Kabanino). *Assets:* They use mostly vanilla civilian clothing or modest gear – we can use mod outfits like additional civilian clothes from Mass's or Windstride's clothing pack. NPCs: expansion traders and some guards at hubs wearing armbands indicating membership. *Missions:* They offer beginner quests ("Help fortify the camp", "Deliver food and meds to another settlement"). They are friendly to players by default. **Theme:** Hopeful rebuilders, represent the "good" side.
- They might reward players with basic supplies or currency for helping.

- As a faction, players can improve reputation with them by doing quests like escorting NPCs or fetching resources.

- **Black Market Collective (Rogues):** This is an underworld faction dealing in contraband (drugs, military gear). They run the hidden black market trader. *Assets:* Perhaps use retextured military gear but with a distinctive look (e.g. black outfits, balaclavas). If a mod like *CJ187's Military Gear* is available, we can outfit them uniquely. NPCs: A black market trader NPC at an outpost, guarded by tough NPCs (who won't attack unless provoked). *Missions:* They might ask players to do dirty work: "steal a supply drop before the Union gets it" or "eliminate an informant zombie in a police station and bring us the data". They toe the line of morality, so doing their missions might lower rep with the Survivor Union (we can have reputation as a system where helping one side angers another, if we want depth). **Theme:** Shadowy opportunists – not outright evil, but self-serving.

- This adds a grey moral area for players: trade safety for profit. But since it's PvE, it's more flavor than actual betrayal mechanics.

- **Military Remnants (CDF Remnant or NATO force):** An organized paramilitary faction occupying certain strategic areas (like Northwest Airfield, or a checkpoint). They could be AI soldiers – heavily armed and hostile by default, viewing all unaffiliated survivors as threats (due to quarantine directives). *Assets:* Use vanilla Chernarus Defense Forces uniforms or a mod that adds Eastern European military outfits. Possibly SNAFU weapons for their loadouts to be formidable. *Missions:* They serve as high-end PvE challenges. For instance, a quest chain might involve discovering that this faction holds the cure or important data, leading to a storyline where players must infiltrate or gain trust.

- We might allow players to *ally* with them eventually (perhaps by doing a very difficult questline assisting them against infected). At that point, their AI could become neutral to those players (hard to implement directly, but maybe we can simulate by having an item or armband that stops them firing at you).

- Initially, though, they are essentially **endgame enemies** patrolling sites like Tisy base. Clearing them could be an objective in the final quests.

- **Scientific Coalition (EGIS, etc.):** A faction of scientists and medics trying to find a cure or understand the infection. They might have a base in a hidden lab (perhaps we add a bunker interior via mods). *Assets:* Hazmat suits (there are mods for Hazmat gear), lab coats, maybe equipment from Namalsk mod if allowed. *Missions:* Centered on gathering data – e.g. collect infected tissue samples, test water sources, escort a scientist NPC to a location. They would be mostly friendly (maybe some are static NPCs that give quests). Their presence drives narrative: e.g. they are the reason certain areas are locked (containment zones).

- We could incorporate **Airdrop events** as "supply drops from the Scientific Coalition" that players can secure [75].
- Players might have to rescue a missing scientist from an infected city as a quest, etc.

- Faction assets could come from mods like *Research uniforms* or use existing gear with armbands.

- **Bandit Gangs (Multiple small groups):** These are not a unified faction but roaming groups of outlaws. They exist primarily as dynamic AI spawns (from the Dynamic AI mod) – basically the "hostile bandits" mentioned [18]. We can name them in lore (e.g. "Wolfpack", "Cannibals", "Rogues"), but mechanically they all just act as hostiles toward everyone. *Assets:* They'll use mismatched gear, some civilian, some low-end military, perhaps distinct colors per gang if we want (one gang wears yellow raincoats, another wears skull masks, etc.). *Missions:* Quests to clear bandit camps (Exp. Quests AI Camp objective) or bounties to eliminate a bandit leader could be offered by Survivor Union or others. They provide constant PvE combat across the map.

- **Missions Integration with Factions:** We will create quest lines that highlight these factions:

- **Main Quest (The Cure):** Given by the Scientific Coalition – involving finding a cure or understanding the infection. Requires cooperating with various factions: e.g. do tasks for the Union to get resources, infiltrate Military Remnants base to get research data, finally open the secret bunker. Completion might "refresh" the world (lore-wise, maybe stabilize the infection for a while).
- **Faction Sub-Quests:** Each major faction can have its own quest chain. For example:
    - Survivor Union's quest line culminating in establishing a new safe settlement (players contribute materials, defend it from a horde event).
    - Black Market's quest line where you earn their trust to access special goods (maybe transporting contraband or assassinating a dangerous AI boss).
    - Military Remnant's quest (if allow friendly outcome): maybe players can choose to help them secure a region instead of fighting them, offering a different ending.
- **Dynamic Missions** (SurvivorMissions) can be tagged with faction flavor by how we present them in radio. E.g. a radio mission might start with "*This is an SOS from a Survivor Union outpost under attack!*" or "*Black Market challenge: first to bring us 5 bear pelts wins a prize.*" – making the world feel reactive.

Realism and Tone: All missions and factions remain grounded: - No magic or sci-fi quests. Even things like "the cure" are approached realistically (collect blood samples, deliver them, defend a lab while a test runs – not waving a wand to cure). - The **tone** is gritty and desperate but with a possibility of hope through cooperation (PvE focus). We avoid fetch quests for totally trivial things – each mission should tie into survival (food, safety, knowledge) or story (discover what happened, etc.). - Use in-world items for lore: e.g. place **notes and letters** (paper items) around that give hints about faction locations or next mission triggers. Perhaps a player finds a **Military orders** document item (we can create a custom note item) which when read, clues them to go to a certain location (kicking off a mission). - **Mission Progression Locked Areas:** As requested, some map areas will be inaccessible or extremely dangerous until missions progress: - Use **Toxic Zones** or locked doors to gate areas. For instance, the NWAF could be blanketed in a toxic gas (needing full NBC gear which is only obtainable after mid-tier quests with Scientific Coalition). So early on, players physically could go there but would die without protection, effectively locking it. - Or put KeyCard doors on key buildings – e.g. the Green Mountain military bunker (just an example) locked until a keycard is gained. - The **soft gate** approach: make the area full of powerful AI (like Military Remnants at Tisy). New players will be cut down if they go alone; only after gearing up and maybe gaining allies via quests can they tackle it. That is an organic lock.

Persistent Missions & World Changes: We can script certain missions to have one-time effects on the world: - For example, if a quest is to **secure a town and set up a safe zone**, once completed, we can actually spawn friendly NPC guards there and make it a new safe trading spot. This shows player impact. Technically, upon mission completion, we could have the server (via the microservice or Expansion quest completion action) modify some config or spawn list to add those NPCs on next restart. - Another example:

completing “The Cure” quest could temporarily reduce the overall zombie spawn rate (simulating that the infection was slowed). Our microservice can listen for that completion and then edit events.xml or types.xml values for infected (for the next restart) – giving a sense of achievement that actually changes gameplay for a while (maybe slowly creeping back up, which could lead to seasons of content).

Faction Management: We’ll maintain a simple reputation tracking outside the game (in the microservice DB). Expansion doesn’t natively do multi-faction rep, so our approach: - When a player completes a faction quest or kills a faction NPC, we log that and adjust rep. - We’ll output a Discord command (`/faction status`) for players to see their standings. - High rep could unlock a unique trader or better prices. Low rep with Survivor Union might cause their safe zone guards to not allow you in (extreme case). - Since implementing AI behavior changes based on rep is complex, we might not enforce it in game except via quest availability (e.g. if you helped Black Market a lot, Survivor Union quest-giver might not give you their final mission – just an idea).

Group and Solo play: Missions will be doable solo, but some are meant for groups (we will label them as such). Because it’s PvE, we anticipate organic cooperation. We will integrate **Expansion Groups** mod so players can form in-game parties and share quests [24]. The group quest sharing feature means if one member does the objective, all get credit [24], which is great. Our Discord will also facilitate grouping by having a channel for missions posting (the microservice can announce “X mission started, recommended 3+ players”).

Examples of Specific Missions:

- *“First Aid” (early quest)* – given by Survivor Union medic. **Objective:** Craft 3 bandages and deliver to the local clinic. (Teaches crafting rags, importance of medical). **Reward:** Basic backpack and some medicine, plus slight rep with Union.

- *“Hunt or be Hunted”* – radio mission. **Objective:** A pack of wolves is terrorizing an area (AI wolves spawned), kill them all. **Reward:** Some pelts (can sell) and a random hunting scope.

- *“Black Box”* – given by Black Market. **Objective:** Recover a flight recorder black box from a helicopter crash site guarded by infected (spawn a crashed heli + some military infected or an AI squad). Possibly requires a **punch card** to open the crash crate (tie-in with keycard mod). **Reward:** A rare weapon or item, and Black Market rep.

- *“Convoy Ambush”* – dynamic event. **Objective:** A heavily armed bandit convoy (a group of AI in a truck, perhaps simulated by spawning them on a road) is moving through. Players must stop them. **Reward:** The convoy’s loot (which could include vehicle parts, weapons). This uses Expansion AI patrol with waypoints.

- *“Patient Zero” (main storyline climax)* – given by Scientific Coalition after many steps. **Objective:** Enter the underground laboratory (previously locked), fight through infected specimens (maybe special stronger zombies), and either retrieve data or neutralize a mutated boss infected (we might simulate a boss by a zombie with very high health or an Expansion AI “Juggernaut” if available). **Reward:** The “cure” item (which might be just an item of symbolic value) and completion of story. After completion, perhaps global server event triggers: e.g. reduce zombie spawns for a while or unlock a new trader with advanced gear as a thanks.

All missions maintain realism: even the “boss fight” is just a unique infected, not a monster from another world. No aliens, no supernatural.

Mission Implementation Considerations:

- Use Expansion Quests for anything that is multi-stage or requires tracking progress per player (so they can log off and continue later).
- Use Survivor Missions for on-the-fly tasks (they have immediate markers or descriptions and expire if not done in time).
- Use server scripts or direct RCON commands (via microservice) for very dynamic things if needed (e.g. triggering an AI group spawn at a random time outside of those frameworks).

We will document all mission JSONs and scripts in the GitHub, so it’s clear how things are set up.

4. Server Economy Design and Balancing

The server economy is a living system encompassing loot spawning, trading, and resource sinks. Our design ensures **scarcity and realism**, while using AI-driven adjustments to prevent inflation or stagnation.

Loot Tables (types.xml): As mentioned, we’ll fine-tune spawn rates: - Common survival items (food, water purification, tools) spawn enough to support players but not so much that they can ignore hunting/farming. - High-tier weapons and gear will be extremely rare in the wild. E.g., SNAFU weapons like automatic rifles might have nominal = 1-3 across the map, or only appear in contaminated zones or via missions. - We categorize loot into tiers and ensure that Tier 4 loot (NVGs, .308 and above rifles, military vests) spawns primarily in **locked areas or events**. I.e., you get them from heli crashes, boss missions, or trader. - Seasonal variation: we can modify types.xml or events based on real seasons (since long-term server). For example, in winter months we could slightly increase spawn of warm clothing and decrease food (to simulate winter scarcity). Conversely summer might have more food spawn (crops). We can achieve this by keeping separate loot XML profiles and our microservice can swap them out during a major “season update” (with a server restart).

Trader Setup: Using Expansion Market, we will set up **trader categories**: - Categories: Food/Water, Medical, Tools, Ammo, Weapons (broken down by tier), Attachments, Clothing, Building Materials, Vehicles, and Misc (like cassette tapes or rare collectibles). - **Dynamic Pricing Mechanism:** While Expansion Market itself has limited dynamic pricing, our **AI economy microservice** will handle it by analyzing logs: - It will track what items players sell or buy the most. For instance, if players flood the trader with wolf pelts (maybe too easy to farm), the microservice will detect high stock and could lower the sell price for pelts next restart (and possibly lower buy price too, making it less profitable). - Conversely, if an item is rarely found but high in demand (say, antibiotics), and always sold out, the agent might increase the price to reflect value or spawn more via events. - This can be done by editing the Market JSON files or via RCON commands if Expansion offers price adjustments in real-time (likely not, so probably it writes to config for next cycle). - TraderPlus mod had explicit dynamic pricing feature (stock influences price) [44] . If we do use TraderPlus or mimic it, we ensure: “More stock = lower price” [44] , meaning the economy behaves with supply and demand. - **Currency:** We will use a physical currency item (likely the vanilla “NylonSheet” renamed to something or a modded currency). We could name it “Old Rubles” or similar. 1 currency unit’s value and weight will be decided – maybe 1 currency = 100 DayZ rubles, and items cost e.g. 50–1000 currency. If using TraderPlus, multiple currencies could be used (silver, gold) [43] , but to keep it simple, one currency plus maybe trade-only items (like the black market might accept **rare items as currency** for some things; e.g. exchange 10 Cannabis for 1 rare attachment, encouraging drug farming as a mini-economy). - **Item Prices:** Will be set based on real-world scarcity and utility: - Food: moderately priced. A can of beans maybe costs a small fraction of a weapon. But if food is too cheap, players might just buy instead of hunt – we want hunting to be attractive. So perhaps make canned food somewhat pricey, encouraging players to fish or farm which is "free". - Ammo: relatively expensive, especially for high calibers. This makes players value each bullet (and consider crafting ammo if using a mod for it). - Medical: expensive. Health is invaluable in survival, so things like morphine, antibiotics cost a lot or are only at black market. - Weapons: tiered. A basic shotgun or Mosin might be affordable for a new player after doing some small missions. But an M4 or SVD should be a long-term goal or mission reward. Possibly not even sold by regular traders; maybe only black market at high cost. - Attachments: Scopes, suppressors – these can be very useful, so they'll either be rare loot or expensive purchases. - Building materials: We do want players to be able to base-build, so nails, wood, metal should be obtainable. We can sell nails at traders cheaply to avoid bottleneck, or ensure they spawn adequately. - Vehicles: Priced very high. Also possibly require a "purchase license" quest. E.g. to buy a helicopter, you must have completed the Pilot Course mission (we can literally give a "Pilot License" item to the player, which the trader could require in inventory via Expansion's quest unlock feature [76] – Expansion Personal Storage can lock an item behind quest completion, maybe similarly we can lock trader entries behind a quest). - Example pricing scheme: *Mosin* = 500 currency; *AKM* = 5000 currency; *NVG* = 8000; *Helicopter* = 50000 currency. If zombies drop currency in missions or players earn by selling loot, it might take quite some playtime to get 50k – which is desired for longevity. - **Trader Stock Limits:** We will use the stock system so traders aren't infinite: - E.g. the Food trader might only have 10 cans of beans per restock. If players buy them all, no more until server restart (or periodic restock event). - This encourages players to also trade among themselves or go loot. We can even encourage a player-driven economy: e.g. allow players to setup a **community trading post** where they can barter (though with 20 players it's small scale). - Black Market stock will be very limited (e.g. 1 unit of each high-tier gun available at a time). So not everyone can gear up simultaneously with the best weapons.

Dynamic Loot & Events: The economy is also impacted by what spawns in the world: - **Airdrops:** Using an Airdrop mod or Expansion's airdrop feature, we'll have random supply drops that contain valuable goods. These act as periodic injections of rare items into the world, but contested via PvE. The AI microservice can schedule these if needed ("if players are short on medical supplies, trigger a Red Cross airdrop event"). - **Zombie loot:** We could enable zombies to very rarely drop useful items (like a 1% chance of a tool or ammo) – but carefully, to avoid farming exploitation. Maybe only special infected (like military infected drop a random magazine occasionally). - **Mission Rewards:** As designed, missions will often give items or currency as rewards. This is a controlled way to distribute high-tier loot for effort, rather than random chance. E.g. killing a boss zombie might always drop a rare dogtag item that can be traded for a weapon.

Resource Sinks: To prevent runaway accumulation: - **Base maintenance costs:** We can simulate this by requiring consumables to maintain base. For instance, require a **"Repair Kit"** item to refresh a base part's lifetime every few days. These kits cost money or materials. If players don't, the base will decay (sink for building materials). - **Vehicle fuel and parts:** Fuel is finite, so players will spend time or money acquiring it. We could even have the trader charge for refueling services (with a script, but simpler is players trade for jerry cans). - **Weapons wear:** ensure weapons and suppressors wear out with use (vanilla has that). So players will either need repair kits (uses up kits) or buy new ones eventually. - **Death penalty:** On death, players might drop all currency they carry (so if they don't bank it in a base stash or something, it could be lost). If their group can't retrieve it, that money is effectively out of the economy. We might also tie this to an AI mechanic: maybe roaming bandits loot bodies including money, and if not killed, that cash is gone. -

Inflation control: If for some reason wealth still accumulates too much (players sitting on piles of cash), the AI economy agent might respond by raising trader prices globally (simulate inflation), or even spawning an event like a **market crash** – e.g. “The Trader’s safe was robbed by bandits, prices double until it’s recovered!” – initiating a mission to fix it. This is dynamic storytelling to justify economy tweaks. - **Patreon/ Supporter perks:** If applicable (though not asked, but just in case), any IRL donations might be rewarded with cosmetic or slight perks, but not money injection. We avoid pay-to-win injection of funds.

Crafting & Progression: We encourage crafting: - Using mods, allow crafting of bullets (there’s a mod for ammo crafting), medicines (combine herbs for natural cure?), or weapons (maybe craft bows or improvised guns). This provides alternative to buying. - Possibly implement a **skill-based crafting unlock**: For example, only a player with advanced Survival skill (Terje skill) can craft a ghillie suit or plate carrier repair. This indirectly creates a tech tree.

Data Export (CSV/JSON): We will maintain the economy data in a structured way (for ease of adjustments): - A **Baserow** or Google Sheets will have tables for Trader prices and loot spawn counts. For example, a table “trader_items” with columns: class name, category, buy price, sell price, stock, trader location (if not global). This can be exported to JSON to feed Expansion Market. - Another table for “loot_table” with class, nominal, tier, etc., which we can import to types.xml format (there are tools like DZconfig or we can script CSV->XML). - This makes economy tweaks relatively quick: e.g. if we notice something off, update the sheet and regenerate configs for next restart.

We will treat the economy as **“AI-driven”** in that the microservices (Section 6) will continuously monitor and adjust: - e.g. If too many nails in the economy (everyone built bases and sells extra nails, price crashes), the service notes that and perhaps reduces how many spawn or lowers trader buy price (so players won’t get much from selling nails). - Conversely if no one has nails and bases are stagnating, maybe an NPC trader comes through with a nails shipment event (supply injection).

Ultimately, by combining controlled scarcity, dynamic adjustment, and multiple ways to obtain items (loot, trade, missions, crafting), we aim for a **balanced, long-term economy** where players always have something to strive for and resources circulate.

5. Discord Integration & Automation

Our server will be tightly integrated with a Discord community to enhance communication, administration, and even gameplay via bots and AI agents. Here’s how we’ll set up Discord integration:

Discord Bot Setup: We will develop a custom **Discord Bot** (using Python or Node.js) that connects to our DayZ server and microservices. This bot will have multiple functions: - **Status and Notifications:** It will listen for server events (via log webhooks or API) and post messages in Discord channels. For example: - Server restart countdowns (“Server will restart in 10 minutes – get to safety!”). - Dynamic events (“⚠ A helicopter crash has been reported near Gorka!”). - Killfeed: every player death or notable kill gets posted (“ **PlayerName** was killed by Infected (Headshot) at Novaya Petrovka” or “ **PlayerName** killed 12 zombies in Vybor”) – with formatting to distinguish PvE kills, etc. Using Battleye RCon logs or server logs, we can parse kill messages and send them [77]. There are existing solutions like Killfeed bot services [77], but we’ll have our own for customization. - Mission announcements: when a Survivor Mission starts, broadcast it (“📻 New radio mission: Medical Airdrop at Polana factory!”) so even offline players on Discord know what’s happening. - Faction war updates: if we ever have AI faction territory fights, could post results ("Bandits have taken over Stary Sobor!"). - **Commands for Players:** Using Discord **slash commands** or prefix commands, players can query information or even trigger actions: - `/status` – the bot replies with current server population, uptime, and next restart time. - `/killstats <player>` – returns that player's PvE kill counts (zombies, animals, AI) and maybe deaths. We'll maintain a stats database for this. - `/market prices [item]` – the bot looks up the trader price of an item or top 10 price list and posts an embedded message with the data. This pulls from the central economy data (which our microservice can expose via API). For example, a user could see "Rifle Ammo: Buy $100, Sell $30 each; Stock: 50" in Discord without logging in [42]. - `/faction status` – the bot PMs the user their reputation levels with each faction. - `/mission request <type>` – Possibly allow players to request a mission. For instance, a player could trigger a custom mission generation if things are quiet. This would call our mission microservice (maybe gated by cooldown or role so it's not abused). The agent might then generate a personalized mission (like AutoGPT generating a scenario) and inject it into the game (or schedule it). - `/report bug <description>` – This will take the user's input and automatically create an issue in our GitHub repo (via GitHub API), logging the bug report for developers. It can also tag it with the user's name and possibly any relevant server info (we can attach the latest server log snippet if the user says "just happened now"). - `/support` or `/help` – lists available commands and how to contact admins (though we aim for adminless, the bot itself is the "admin" interface). - **Admin Commands:** For server admins (or the AutoGPT agent), some commands may be restricted to a Discord role: - `/server restart` – sends a restart warning and issues an RCon command to restart (this is backup if our scheduled restart fails or manual needed). - `/spawn item <item> <count> <player>` – to give compensation or event rewards. - `/tp <player> <location>` – if needing to rescue a stuck player, etc. Ideally seldom used. - These commands will require a role check so only authorized or the automation agent can use them.

Event Webhooks: We will set up the DayZ server to output logs that our system can read. DayZ doesn't have built-in webhook, but we can tail the server's log file using a script. The script then sends specific events to Discord: - For killfeed, since DayZ writes to a server log or admin log with kill info ("Player X killed by Y"), our log-tail service catches those and posts to Discord. (There are tools like CFTools or existing bots that do this [78] [77]; we can either use their API or do our own via RCON). - For server messages (like "Restart in 10 minutes"), we can either manually create that schedule to Discord or catch in logs if we print it in server announcements. - We will also send certain **microservice events** to Discord via webhook: e.g. the economy agent might post a weekly "State of the Economy" summary in a #server-economy channel (like "Total currency in circulation: 50000; Most traded item: Wolf Pelt; Inflation rate: 5% this week"). - The AI faction controller could post "faction news" ("The Survivor Union strengthened defenses at Green Mountain").

AutoGPT & AI Agents Integration: This is an innovative aspect: - We plan to use an **AutoGPT agent** (or similar autonomous AI) running continuously with access to server data. This agent will have goals like *"Maintain server balance"*, *"Create dynamic missions when players are idle"*, *"Assist moderators"*. The integration loop is: - The agent can receive *triggers* from the log events (e.g., "several players died at one location -> maybe spawn a rescue mission"). - It can output *actions* via either RCON commands or by instructing our microservices. For example, AutoGPT might decide "increase food spawn slightly because many players are starving". It would then call the economy service API to adjust types.xml values or just suggest it to admins. - It can also respond on Discord if asked. We could have a persona for it, e.g. `/ask ai <question>` and it uses game data to answer ("Which areas are safest right now?" The AI checks the last hour of kills and says "The south coast is relatively safe, but avoid NWAF due to high infected activity."). - **Discord → AutoGPT → DayZ loop example:** An admin types `/mission create scavenger`. The Discord bot passes this to the

AutoGPT instance or a specialized Mission Generator AI. That AI builds a mission JSON (maybe using templates and randomization), the mission microservice injects it into the Survivor Missions or Expansion Quest system (e.g., adds a new quest available and notifies players). The Discord bot replies "Mission created: Scavenge the Crash Site at grid 045 120". - **DayZ → AutoGPT loop example:** The server log says "Animal population very low" (could be from our own monitoring). AutoGPT sees that and decides to increase deer spawn by editing events.xml for next restart, and posts on Discord "Wildlife populations seem depleted, adjusting ecosystem to compensate."

We must ensure the AI doesn't do anything too crazy without oversight. Likely we'll run it in a sort of *"propose action then execute"* mode. For critical changes, it could open a GitHub Pull Request (for config change) and notify admins to approve. This is where the GitHub SDLC comes in (section 7).

Discord Role Sync: If we have supporters or admins, the bot can link game whitelist or give titles: - We could implement a system where Discord roles (e.g. VIP) correspond to an in-game armband or a slight perk (like base building limit increased). Achieved by maintaining a Steam64 <-> Discord mapping (players submit their Steam via a command or we use Steam sign-on through a web form). - For Patreon integration: the bot can detect if a user has a Patreon Supporter role (manually given or via Patreon API) and then grant them something in-game – maybe a custom skin or their base gets an **insurance** (not wiped as quickly). We must keep it non-pay-to-win.

"Adminless" Moderation: Instead of active admins, we rely on logs and bots: - The bot monitors chat logs for slurs or toxic language (if any, since mostly PvE coop, but just in case) and can issue warnings automatically. - If a player is caught exploiting (e.g. glitching through walls, which might be detectable by unusual position logs), the bot could flag it for review. - Use **BattleMetrics** or a tool to detect known cheater accounts by GUID and auto-ban them (though being PvE with mods, hacking is still possible and could ruin PvE by spawning stuff – we have to remain vigilant). - The Discord can serve as a knowledge base: we maintain FAQ answers the bot can provide if players ask common questions.

Integration Methods Recap: - **Battleye RCON:** We will use a library (like node-battlEye or python Battleye library) to allow the bot to send commands to the server (e.g. say messages, kick, etc.) [79] . We'll secure this by IP whitelist or running the bot on the same machine if possible. - **Log Tailing:** A small Python script that reads the DayZ server log file (or named pipe) in real-time and sends events to a message queue or directly to Discord via webhook. This is especially for killfeed and certain event triggers since RCON does not output all logs. - **REST API Microservice:** We might create a microservice with a REST API that can be polled or triggered. For example, the Discord bot calls `GET /api/prices?item=AKM` on our Economy service, which returns JSON of current price, which the bot then formats to Discord. Similarly for faction status queries. - **Webhooks from Game to Bot:** If using CFTools Cloud or equivalent, there are ways to have game events forwarded, but we prefer self-hosted. Possibly, we implement a small in-game script (if modding the mission) that does an HTTP POST on certain events to our bot's endpoint. But since DayZ scripting can't directly call web APIs, we likely stick to external monitoring.

Diagrams: To illustrate, here's a high-level integration flow:

```
flowchart LR
    subgraph DayZ Server
      A[DayZ Server Instance] -- logs --> B(Log Tailing Agent)
      A -- BattlEye RCON --> C(RCON Bot Service)
    end
    subgraph Discord
      D[Discord Client Users] -- commands --> E(Discord Bot)
      E -- notifications --> D
    end
    subgraph AI Microservices
      F(Economy Service)
      G(Mission Generator AI)
      H(Faction/Story AI Agent)
    end
    B -- event JSON --> H
    B -- kill events --> E
    C -- server status --> E
    E -- slash API calls --> F
    E -- mission request --> G
    H -- decision (e.g. adjust prices) --> F
    H -- decision (spawn event) --> G
    G -- mission data --> DayZ Server
    F -- market info --> DayZ Server & DB
    F -- report/confirm --> E
```

In this diagram: - The Log agent sends events to the Faction/Story AI (H), which might decide to create a mission via G or adjust economy via F. - Discord bot communicates both ways with players and the services.

And a **sequence example** for mission creation via Discord:

```
sequenceDiagram
    participant User as Discord User
    participant Bot as Discord Bot
    participant AI as Mission AI Service (AutoGPT)
    participant Server as DayZ Server

    User->>Bot: /mission create rescue
    Bot->>AI: "Create a rescue mission" (API Call)
    AI->>AI: [Generate mission JSON: location, enemies, objectives]
    AI->>Server: [Inject mission data] (via SurvivorMissions or Expansion Quest)
    Server-->>User: (In-game) Radio: "New Rescue Mission available!"
    Bot->>User: "Rescue mission created at Grid 045 120 (Check your radio)"
```

This shows how the pieces connect on a command.

Community Engagement via Discord: Beyond commands, we'll use Discord for community building: - Channels for **lore** (we can post short stories or newspaper clippings to enhance the narrative). - **Leaderboards:** The bot can post daily or weekly leaderboards (most zombie kills this week, longest survival time, etc.) to a channel. - **Suggestions/Voting:** Perhaps players can vote on events ("Next weekend event:

horde night or double resources?”) with polls. - **Moderation Logs:** A private admin channel where the bot logs any suspicious activity or admin commands executed for transparency.

All in all, Discord becomes an extension of the game: - Players online and offline remain connected (someone at work can check Discord to see if an event popped and coordinate with others to join in the evening). - It reduces need for in-game UIs for some info (prices, etc., are available on phone via Discord). - Serves as a **bridge for our AI** – making it easier to deliver AI-generated content to players (since formatting things in-game is harder, the AI can always dump richer info in Discord if needed, like a story of what happened in a mission aftermath).

We will document how to use all these features in a Discord pinned message or a web README so players aren’t lost.

6. AutoGPT Microservices Architecture

To implement the adaptive and intelligent server features, we'll design a set of **microservices and AI agents** working in concert. This modular architecture allows each aspect (spawns, missions, economy, etc.) to be managed, monitored, and improved independently. Below are the key components:

6.1 Microservice Overview:

- **Log Tailer / Event Publisher:** A lightweight service (could be built in Python) that continuously reads the DayZ server logs (and/or listens to RCON) and filters important events. It then publishes these events to a message bus or directly triggers other services. For example, on a kill event, it sends a JSON like `{"type": "kill", "victim": "Player1", "killer": "Zombie", "location": [x,y]}` to a central queue. On a player connected event, or server started, it sends those too. This acts as the **eyes and ears** of the AI system within the game.

- **RCON Controller Service:** Manages direct RCON communication for command execution. It can be invoked by other services to run admin commands (spawn items, send messages, etc.). It also periodically queries server status (player count, etc.) for the Discord bot. This might be part of the Discord bot process or separate.

- **Spawn Controller Service:** Responsible for dynamic spawns outside of static events.xml. For example, if we want to spawn an AI horde or a custom vehicle at runtime, this service takes commands (from AI or triggers) to spawn entities. Since DayZ doesn’t expose an API, spawn has to be via existing mod hooks (like using SurvivorMissions to spawn AI, or injecting an object spawn via a mod). Alternatively, some spawns can only happen on restart by editing XML files. In those cases, this service can edit the relevant XML/JSON and flag the server to load it on next restart. For example, to spawn a new car, it could add an entry to cfgeventspawns.xml and then either require restart or use admin command to create vehicle (if a mod provides a function).

- This is more of an orchestrator for the environment. It could also handle weather if we script custom weather patterns (DayZ has random weather but maybe we want story-driven storms – we can use RCON to change weather states via commands if available).

- **Mission Generator AI:** This is an AI-driven service (could incorporate an LLM like GPT-4, fine-tuned or via API) that creates mission scenarios. It could have a prompt like: *"Generate a JSON mission where objective is X, ensure it's realistic and fits current server context Y"*. The service would integrate with our mission framework. For example:

- For SurvivorMissions, it might output a new mission file (in the format that mod expects, e.g. an .json or .sqf snippet).
- For Expansion Quests, it could directly modify the Quest config and add a new quest entry, although expansion quests might need a restart to load new quests – so maybe SurvivorMissions is better for truly dynamic ones.
- We can run this AI either continuously or on-demand (from a Discord command or triggered by e.g. "no missions in last hour, players bored -> generate one").
- Importantly, we'd restrict its creativity to avoid lore-breaking stuff. Possibly by providing templates or requiring it to pick from pre-vetted objectives and locations (we can supply it a list of possible locations and enemy types).

- This AI can be part of the AutoGPT agent or a separate unit that focuses on missions only (for safety, since mission generation is complex, a dedicated process is safer).

- **Economy Manager AI:** This service monitors all economic transactions and world loot metrics. It could be simpler (algorithmic adjustments) but we want AI logic too:

- It keeps track of how much of each item exists (we can count items on server via logs or by periodically scanning the storage files – though that's heavy. Alternatively, hook into trader logs: each sale/purchase is logged so we know flow).
- If it sees imbalance (like too much money in circulation or one item becoming meta), it can suggest or apply changes. E.g., "556 ammo is being bought out constantly -> increase price by 20% next restart" or "nails stockpiling -> spawn fewer in loot".
- It could use a small rules engine: For instance, maintain target ratios of certain item usage, or ensure overall trader stock value doesn't exceed some threshold.
- Ideally, it notifies via Discord its decisions: "Economy AI: Adjusting prices of medical supplies due to shortage [42]."

- Implementation: likely not an LLM for the actual math, but an LLM could be used to explain decisions or consider more abstract aspects (like reading a weekly summary and deciding "players are not spending money, perhaps introduce a gold sink event"). But to start, algorithmic approach with some configurable parameters is enough, with final decisions approved by an admin or just executed if low-risk.

- **Factions & Story AI:** Possibly the most interesting – an AI that "narrates" the world's ongoing story. This could be an AutoGPT agent with goals like:

- "Keep the world interesting by triggering faction conflicts or events occasionally."
- "Make sure each faction feels alive by sending radio messages or Discord lore snippets."
- "When players accomplish major tasks, advance the plot accordingly."
- It would take input like faction reputation levels, mission completions, maybe even player locations, and then decide on world changes. For example, if players have been ignoring the Black Market, the

AI might decide “Black Market tries to entice players with a special sale event” and triggers a special inventory item for sale or a Discord message from a Black Market NPC.
- It could also handle **dynamic storytelling**: generating a short news bulletin each day summarizing what happened (this might read logs like a newspaper: “Day 5: A group of survivors cleared out Vybor, and a mysterious helicopter was heard overhead in the north…” then post that to Discord or in-game note).
- Technically, this agent can use an LLM to generate text (for radio messages or NPC dialogue) to keep things fresh. We must moderate outputs (maybe keep it on rails with story prompts).

- It might also instruct other services: essentially it’s the high-level brain that might say “spawn more bandits in the south today because it's been quiet there” (then spawn service does it), or “the Military Remnants are retaliating tomorrow” (schedule a big event).

- **GitHub SDLC Agent:** We want a robust deployment pipeline, so this service (or just a manual process) will manage code and config through Git:

- All configs (xml, json, mission files) are stored in a GitHub repository. When we or the AI make changes, they get committed. Possibly the AI can even commit (with descriptive message like “Economy tweak: raised ammo prices”).
- We can set up a **CI pipeline** (like GitHub Actions or Jenkins on our server) that upon new commit, will package and deploy the changes to the server (or if it’s just config, simply place them and schedule a restart).
- For mods and binaries, we likely still update manually via SteamCMD, but config and scripts can be automated.
- The Discord `/report bug` we set up goes to GitHub issues – linking community feedback directly into our dev process.
- Also, documentation and balancing data can live in the repo, making it easy to track changes over time (e.g. see how loot values changed from Season 1 to Season 2).

Architecture Blueprint: The system is event-driven and modular, loosely following a publish-subscribe model: - The Log Tailer is a **publisher** of game events. - Various **subscribers** (AI services) consume those events and may output new events or actions. - We could use a simple message queue (like RabbitMQ or Redis pub/sub) if needed to decouple, but given scale, even direct HTTP calls from the log parser to services might suffice.

Sequence Example (Comprehensive): A day in the server: 1. **Morning:** Server restarts at 6 AM. Economy Manager service has pre-written updated `MarketPrices.json` based on yesterday’s trends and GitHub-deployed it before restart. Server comes up with new trader prices and maybe adjusted loot quantities. 2. **Gameplay:** Players play, log events stream out. Log Tailer sends events: - Player `Alice` kills 10 zombies in Elektro → Bot posts killfeed, Faction AI notes “Alice is doing good in south”. - A dynamic mission completes successfully → Mission service tells Faction AI which faction was helped. - A player buys out all antibiotics from trader → Economy manager notes it. 3. **Midday:** Faction AI sees no major events for a while and decides to spice things up: - It sends a command to Mission Generator: “create a mission for Black Market faction, easy difficulty, in area where players are (south perhaps)”. - A mission is generated and started – Discord and in-game get notifications. 4. **Economy:** Many wolves have been killed (maybe players farming pelts), Economy AI sees 50 wolf pelts sold in 3 hours, triggers: - Post a message “Market adjusting: wolf pelt price decreased due to oversupply.” - Writes change to config for next restart. - Possibly also triggers an event: maybe spawn fewer wolves via spawn service to prevent over-farming (or conversely, spawn more wolves to fight back? Could go either way). 5. **Evening big event:** Players finally complete the big "Clear Military Remnants" mission. This triggers: - Mission service flags it, Faction AI receives it. - Faction AI decides story advancement: Maybe it says "Military Remnants defeated at NWAF!". It instructs Spawn service to stop spawning that AI at NWAF (so players can loot freely for a bit). It tells Mission AI to generate a follow-up quest for retrieving intel from their base. - Also Faction AI uses Discord to narrate: "Radio: ... the rogue military forces at the airfield have been neutralized by brave survivors!" - The completion of this quest might also adjust faction relations (players who participated get +rep with Survivor Union, etc.). 6. **Admin oversight:** Throughout, logs are being recorded. At day's end, the SDLC pipeline might compile a summary of config changes and events. Admin/dev can review GitHub commits made by AI (if any) to ensure nothing unbalanced. - If something went wrong (say the AI did an economy change that broke something), we roll it back via Git easily and maybe tweak the AI's rules.

The key is that *each microservice has a clear domain:* missions, economy, factions, etc., and they communicate through clearly defined events or APIs. The AutoGPT (or chain-of-thought agent) can either be one monolithic agent handling multiple goals or separate specialized agents (one for economy, one for story). Given complexity, starting with separate logic for each might be more predictable, but a more advanced approach might use a single AutoGPT that can call different "tools" (our microservices are effectively tools for it: one to change economy, one to create mission, one to send discord message). This is how AutoGPT often works (with plugin-like tool usage) [80] [81] .

Technology stack: - Python for quick scripting of log tail, using something like Watchdog to detect log writes. - Node.js or Python for Discord bot (discord.py or discord.js libraries). - The AI models could run via OpenAI API (GPT-4) or local (if we want self-hosted later). Initially using API might be simpler for quality results, but we will carefully limit when it's called (some mission text generation, story blurbs – not every second). - Database: likely just JSON files or in-memory for low scale (20 players). But we might use SQLite or simple file storage for things like player stats and rep so it persists across restarts. - Each service in Docker containers possibly, to isolate (makes deployment/updates easier in pipeline).

Mermaid Diagram (Summary):

At risk of repetition, the earlier flowchart covers architecture. We can refine it focusing on microservices:

```
flowchart TB
    subgraph Game Server
      subgraph DayZ
        S1(ServerDZ.exe)
      end
      L[Log Tailer] --> MQ((Event Bus))
      S1 --> L
      R[RCON Client] --> S1
    end
    subgraph AI & Services
      MQ --> MGS(Mission Generator AI)
      MQ --> ECS(Economy/Market Service)
      MQ --> FCS(Faction/Story AI)
      MGS --> S1
      ECS --> S1
      FCS --> MGS
      FCS --> ECS
      FCS --> R
    end
    subgraph Discord & Ext
      DBot(Discord Bot) --> R
      DBot --> ECS
      DBot --> MGS
      DBot --> FCS
      GH[GitHub Repo] -->|CI/CD| S1
      DUser(Discord User) --> DBot
      DBot --> DUser
    end
```

This shows Discord bot interfacing with RCON and services, event bus feeding AI, and GitHub CI deploying to server.

Potential Challenges: - Ensuring consistency: If the AI tries to change something mid-session that requires restart (like economy or adding a quest), it might not see immediate effect. We'll have to sometimes queue changes for restart and communicate that. - Avoiding AI chaos: We'll implement safeguards (for instance, capping how extreme a change can be: AI won't set price of bread to 0 or spawn 1000 zombies; also require certain format so it doesn't break XML). - Testing: We will simulate events in a test environment (maybe feed the AI some fake logs) to see its responses before trusting it live.

In essence, this architecture turns the server into a **self-regulating ecosystem**, with AI as the game master that watches and tweaks knobs to keep things engaging and fair.

7. Documentation and Development Workflow

We will treat this project with the rigor of a software development project, maintaining clear **documentation**, version control, and a structured workflow for updates. The deliverables and practices include:

7.1 Documentation Structure:

All documentation will be written in Markdown for easy viewing on GitHub and in any future web docs. We will maintain a dedicated repository (or a `/docs` folder in the server config repo) for docs, containing:

- **README.md:** A high-level overview of the project, setup instructions for someone who wants to host or contribute, and a feature list. It will explain the server concept, list of major mods, and quickstart guide for new players.

- **/docs/ARCHITECTURE.md:** A detailed description of the system architecture (much of what we've described in Section 6), including diagrams. We will include our Mermaid diagrams or exported images of them so others can visualize the data flow.

- **/docs/CONFIG_GUIDE.md:** Explains each important DayZ config (types.xml, events.xml, etc.) and how we've customized them. It will be sort of an annotated config, e.g., "We set `disablePersonalLight` to true for realism [9], increased `ZombieMax` in globals.xml to 150 to allow more infected," etc., citing any references or rationale. This helps if others (or future us) need to tweak settings, they know why something is set a certain way.

- **/docs/MODS.md:** A list of all mods used with their versions, links to Steam Workshop pages, and any special installation notes or config changes. For instance, note that *TerjeSkills* requires `TerjeSettings/Skills.cfg` present and how we configured it. If any mod needed an override or patch, we document what we did.

- **/docs/ECONOMY.md** (or perhaps in a data folder): A description of the economy system. Possibly include tables or CSVs of current prices and logic behind them. Also instructions how to update prices via Baserow/CSV and push to the server.

- **/docs/MISSIONS.md:** Documentation for mission creators or for QA – listing the custom missions we've implemented (with IDs, triggers). Possibly instructions on adding new Expansion Quests or Survivor Missions scripts.

- **/docs/AI_AGENTS.md:** A guide on the AI components – how the AutoGPT agent is set up, what prompts we use, how to monitor or adjust it. This is crucial because if another developer takes over, they should know how to handle the AI's behaviors.

- **Mermaid Diagrams and Flowcharts:** We will include diagrams in the above docs where relevant. For example, ARCHITECTURE.md will have the architecture diagram; ECONOMY.md might have a flowchart of how money flows; MISSIONS.md might include a state diagram of a quest line. We will likely embed them via Markdown mermaid code blocks or include image files generated from mermaid.

- **Code Samples & JSON Templates:** Provide example config fragments. E.g., in MISSIONS doc, include a snippet of a Survivor Missions config for one mission, to serve as template. Or in CONFIG_GUIDE, show a piece of types.xml for how we set up a new item spawn.

- **Usage Guides for Players:** Possibly a **Survival Guide** we can share with players (in the repository or via a website) explaining server-specific mechanics (like skills system, how to use cassettes, etc.). This might be /docs/PLAYER_GUIDE.md. It would be written in an in-character style maybe, but with clear instructions. This ensures players understand the new features (since it's heavily modded, new players need guidance).

7.2 SDLC (Software Development Life Cycle):

- We will use **GitHub for version control** of configurations, scripts, and docs. Each change is made via pull requests, even if just us and our AI – so that every alteration is tracked. E.g., if the economy AI changes prices, ideally it makes a commit "Adjust trader prices: wolf_pelt -20%" with details. We can review commit history to debug issues (like "why is food not spawning? oh a commit two weeks ago set food nominal to 0 by mistake").

- **Branching Strategy:** Possibly a `dev` branch where experimental changes from the AI or testing go, and a `main` branch always containing the live server config. This way, we can test changes in a local server environment (or a test server instance) before merging to main for live use.

- **Continuous Integration:** Setup GitHub Actions or another CI pipeline to do things like:

- Lint or validate XML/JSON on each commit (to avoid a syntax error bringing down the server on restart). There are XML validators – we can run `xmllint` on config files, or use DayZ modding community tools (DZconfig API can validate types.xml).
- Possibly run a **headless DayZ server test** that ensures it starts up with the config (though starting DayZ server in CI might not be feasible due to no headless mode or license issues).
- At least run Python unit tests for any scripts (like economy adjustment logic).

- If all checks pass, either automatically deploy to server or signal it's ready.

- **Deployment:** We could automate deployment with a script on the server that pulls the latest GitHub main branch and copies files into the server directory, then restarts the server. Or use a tool like Chef/Ansible for config management. Because we have many moving parts (DayZ server, microservices, Discord bot), containerization might be helpful:

- A Docker Compose file could define the microservices and bot, so we can deploy them consistently.
- The DayZ server itself cannot be containerized easily due to Windows dependency (unless using Windows containers, which is complex). But we can run it on the host and just have volumes mount the config from our repo clone.

- The pipeline might simply SSH into the server and run a deploy script (for a self-hosted scenario). We'll ensure security (only allow the CI key limited access).

- **Issue Tracking:** We'll use GitHub Issues to track bugs and feature requests (with players able to report via Discord -> bot -> issues). We'll categorize issues by type (mission bug, economy balance, etc.) and link them to changes in commits. This keeps a history of decisions (e.g., “Zombies too easy” issue -> resolved by commit that increased zombie damage by 20%).

- **Continuous Feedback:** We’ll encourage players to give feedback via Discord `#feedback` channel. The Discord bot might even capture quick feedback polls, e.g., after an event, it could ask “How was this event? (👍/👎)” and log results.

- **Backups:** In addition to Git (configs) and server persistence backups, we'll backup the database of player progress (if any external DB for skills) regularly, and even the Discord bot's data if needed.

7.3 Example File Structure:

We will organize the files clearly in the server directory or repository. For instance, in the repository:

```
/config/
    serverDZ.cfg
    types.xml
    events.xml
    globals.xml
    cfgeconomycore.xml
    ... (other vanilla files)
    MapEditObjects/   (custom map additions in Expansion format or editor
format)
        racetrack.map
        factionbases.map
    ExpansionSettings/
        MarketSettings.json
        AISettings.json
        QuestSettings.json
        MissionFiles/ (if needed)
    ProfileExamples/
        ExpansionMod/Quests/Quests.json (our configured quests)
        SurvivorMissions/ (our custom missions scripts)
    ...
/mods-config/
    BaseBuildingPlus.json
    TerjeSkills.cfg
    SFRadio/ (if any custom track config)
/microservices/
    economy_service.py
    mission_service.py
    faction_ai.py
    discord_bot.js
    requirements.txt (for Python libs)
/docs/
    README.md
    ARCHITECTURE.md
    CONFIG_GUIDE.md
    ... etc
```

This layout separates vanilla config from mod-specific config and code. When deploying, some of those go to the server profile folder, some to mpmissions, etc., which our deploy script will handle.

7.4 Developer Documentation & Onboarding:

We will also document how a new developer or admin can set up the development environment: - Tools needed (VSCode with an XML plugin, Python 3.9+, etc., DayZ client for testing mods locally). - How to run a local DayZ server with the mod pack (perhaps using DayZ Docker or just manual). - How to run the microservices (e.g., `pip install -r requirements.txt` then `python economy_service.py`). - How to get credentials like Battleye RCON password and Discord bot token securely (not in repo, but maybe as environment variables). - Code documentation: in our Python/Node code, use clear function names and comments, possibly docstrings, so even if it's small, it’s understandable.

7.5 Mermaid and Diagrams in Documentation:

We will include all relevant diagrams we crafted above into the docs. For example, in ARCHITECTURE.md, include the mermaid sequence diagram of the Discord command flow, and the flowchart of the microservices. We ensure not to put images in front of headings (per guidelines), but within the narrative, and cite any source if needed (though these diagrams are original).

7.6 Baserow/CSV usage:

We maintain CSV files for things like trader data as mentioned. Possibly store them in a `/data` folder: - e.g., `trader_prices.csv`, `loot_table.csv`. We then have scripts or just manual process to convert them to the needed format. We might automate conversion with a Python script (`generate_types.py` reads loot_table.csv and produces types.xml ensuring formatting and preserving comments). - Keep the CSVs in Git as the source of truth for easier editing by non-coders (could even connect Baserow API to our GitHub via some action if we want to get fancy, but likely just export manually from Baserow then commit). - Document in ECONOMY.md how to update these.

7.7 In-Game Documentation:

We'll also utilize in-game means to inform players: - Possibly create an item like a "Journal" or "Server Guide" book that spawns with players or at spawn points, containing a summary of key server features and a link to the Discord. DayZ allows custom notes or editing the text of books via stringtable mod, we could do that. - The loading screen (if we can mod it) or server Message of the Day (MOTD configured in serverDZ.cfg) will list "Join our Discord for server info and rules" [82], and maybe brief tagline ("Hardcore PvE Survival – AI missions and dynamic world"). - The MOTD can also remind "Use /help on Discord to see bot commands".

Conclusion: With thorough documentation and a robust development pipeline, we ensure the project is maintainable long-term. Every system from configuration to AI is transparent either through documentation or logged output, making it easier to tweak or hand over to someone else if needed. The realism focus remains in all docs – e.g., we note in docs that any future mod or mission must align with the grounded, survival tone (explicitly stating no sci-fi, etc.). This acts as a guiding principle for anyone contributing.

By following these guidelines and processes, we can deliver a **developer-grade** output that not only provides a rich gameplay experience but is also sustainable and extensible.

Sources:

For clarity and verification, the design references several DayZ mod features and recommendations:

- DayZ server configuration keys and example settings [7] [9]
- Hardware requirements and performance tips [2] [6]
- OVH Game server DDoS protection [1] [4]
- Dynamic AI mod description (NPC spawns) [17] [18]
- Expansion Quests features (quest types, AI integration) [23] [25]

- Trader mods and dynamic pricing (TraderPlus features) [42] [44]
- Expansion mod overview (vehicles, base, AI, traders) [39] [40]
- SNAFU Weapons mod aim (realism and variety of firearms) [83]
- TerjeSkills mod (skills and perks list) [55] [57]
- SF Radio mod (working cassette radio) [63]
- Novy Lug Race Track mod info [65]

These ensure our decisions are grounded in existing capabilities of mods and best practices from the community.

---

[1] Rent, Create and Host a DayZ Server | OVHcloud Worldwide
https://www.ovhcloud.com/en/bare-metal/game/dayz-server/

[2] [3] Server specs? : r/dayz
https://www.reddit.com/r/dayz/comments/el057i/server_specs/

[4] DayZ game server : r/OVHcloud - Reddit
https://www.reddit.com/r/OVHcloud/comments/17djc5p/dayz_game_server/

[5] [6] [39] [40] [46] [53] [54] [75] The Best DayZ Server Mods and How to Install Them in 2025 September 2025 | rocketnode.com
https://rocketnode.com/blog/best-dayz-server-mods-2025/?srsltid=AfmBOooXn5R-3pzSkcuNs45O0wIfJklL6bJnLIqYyHMr_eQmI8Kuicop

[7] [8] [9] [10] [11] [14] [82] serverDZ.cfg Configuration - DayZ Server Documentation - DZconfig Wiki
https://dzconfig.com/wiki/serverdz

[12] [13] serverDZ.cfg missing - Servers - DayZ Forums
https://forums.dayz.com/topic/240766-serverdzcfg-missing/

[15] All Dayz Server Settings - indifferent broccolipedia
https://wiki.indifferentbroccoli.com/DayZ/ServerSettings

[16] Steam Workshop::DayZ-Expansion-AI
https://steamcommunity.com/sharedfiles/filedetails/?id=2792982069

[17] [18] [19] Steam Workshop::DayZ-Dynamic-AI-Addon
https://steamcommunity.com/sharedfiles/filedetails/?id=2874589934

[20] Steam Workshop::Zombies-Health-rebalance
https://steamcommunity.com/sharedfiles/filedetails/?id=1633167429

[21] Steam Workshop::WalkingDeadZombies
https://steamcommunity.com/sharedfiles/filedetails/?id=1609324498

[22] How to get AI quest NPC to spawn in DayZ after installing expansion ...
https://www.facebook.com/groups/620596961818260/posts/1553410058536941/

[23] [24] [25] [32] [37] [76] Steam Workshop::DayZ-Expansion-Quests
https://steamcommunity.com/sharedfiles/filedetails/?id=2828486817

[26] guiltysyndicate/Unofficial-Quests-For_DayZ-Expansion - GitHub
https://github.com/guiltysyndicate/Unofficial-Quests-For_DayZ-Expansion

[27] [28] [29] [30] [38] Steam Workshop::Survivor Missions
https://steamcommunity.com/workshop/filedetails/?id=1988925918

[31] SMVPPM - Steam Workshop
https://steamcommunity.com/sharedfiles/filedetails/?id=2268305113

[33] [34] KeyCard-Rooms - Steam Workshop
https://steamcommunity.com/sharedfiles/filedetails/?id=2620165863

[35] Punched Card - DayZ Wiki - Fandom
https://dayz.fandom.com/wiki/Punched_Card

[36] scalespeeder/DayZ-Chernarus-Custom-Bunker-With-Loot-for-PC ...
https://github.com/scalespeeder/DayZ-Chernarus-Custom-Bunker-With-Loot-for-PC-and-Console

[41] Steam Workshop::Black Market - The Architect
https://steamcommunity.com/sharedfiles/filedetails/?id=2908878542

[42] r/dayz - A new trader mod made by me is available for everyone ...
https://www.reddit.com/r/dayz/comments/nkuhlb/a_new_trader_mod_made_by_me_is_available_for/

[43] TraderPlus | DmitriDayZModding - Jimdo
https://dmitridayzmodding.jimdosite.com/traderplus/

[44] TraderPlus - Steam Workshop
https://steamcommunity.com/workshop/filedetails/?id=2458896948

[45] H3liumCat/SNAFU-Expansion-Trader-Config - GitHub
https://github.com/H3liumCat/SNAFU-Expansion-Trader-Config

[47] DayZ-Expansion-Vehicles - Steam Workshop
https://steamcommunity.com/workshop/filedetails/?id=2291785437

[48] [49] [50] [51] [52] Steam Workshop::RedFalcon Flight System Heliz
https://steamcommunity.com/workshop/filedetails/?id=2692979668

[55] [56] [57] [58] [59] [60] [61] [72] Steam Workshop::TerjeSkills
https://steamcommunity.com/sharedfiles/filedetails/?id=3359678303

[62] Steam Workshop::MedicalAttentionUpdated
https://steamcommunity.com/sharedfiles/filedetails/?id=2265389602

[63] [64] Steam Workshop::sf_radio
https://steamcommunity.com/sharedfiles/filedetails/?id=2517480838

[65] [66] [67] [68] [69] [71] Steam Workshop::Novy Lug Race Track
https://steamcommunity.com/sharedfiles/filedetails/?id=2929042272

[70] DayZ-Expansion-Bundle - Steam Workshop
https://steamcommunity.com/sharedfiles/filedetails/?id=2572331007

[73] [74] [83] Steam Workshop::SNAFU Weapons
https://steamcommunity.com/sharedfiles/filedetails/?id=2443122116

[77] Killfeed.DEV | DayZ Killfeed
https://killfeed.dev/

78 CarimDayZ/carim-discord-bot - GitHub
https://github.com/schana/carim-discord-bot

79 DayZ: RCON | ZAP-Hosting Docs
https://zap-hosting.com/guides/docs/dayz-rcon/

80 What is AutoGPT? Complete Guide to Building AI Agents
https://www.codecademy.com/article/autogpt-ai-agents-guide

81 Deep Dive into AutoGPT: The Autonomous AI Revolutionizing the ...
https://peter-chang.medium.com/deep-dive-into-autogpt-the-autonomous-ai-revolutionizing-the-game-890bc82e5ec5