Cryptoverse

1.Introduction

Team Members

M.Saranya

K.Sathya

G.Rubasri

D.Sandhiya

2.Project overview

Cryptoverse is an immersive 3D virtual world built using Unreal Engine 5, developed by the team behind ChainGuardians. This metaverse platform enables users to create, own, govern, and monetize experiences using the native token, $CGG.

Key Features:

- **Diverse Zones:** The Cryptoverse is structured into eight distinct zones, each offering unique aesthetics and functionalities tailored to various activities such as business, entertainment, fashion, and more.
- **LAND Ownership:** Users can acquire virtual real estate, known as LAND, represented as NFTs. These LAND parcels can be bought, sold, or rented, providing opportunities for users to develop and monetize their virtual spaces.
- **Prana System:** This innovative feature rewards user engagement and interactivity within the virtual world. A higher Prana Score grants access to exclusive features and rewards in the platform's marketplace.
- **Cross-Reality Experiences:** Cryptoverse integrates augmented reality (AR), virtual reality (VR), and mixed reality (XR) to offer immersive shopping experiences, virtual store solutions for businesses, and phygital assets that bridge the gap between physical and digital worlds.

- **User-Centric Tools:** The platform provides flexible building tools suitable for users with varying levels of programming expertise, facilitating the creation and customization of virtual spaces.

3.Architecture

Cryptoverse's architecture is designed to provide a highly immersive and scalable metaverse experience. It is built using **Unreal Engine 5**, ensuring high-fidelity graphics, realistic physics, and seamless real-time interactions. Below are the key components of its architecture:

1. Core Infrastructure

- **Unreal Engine 5:** Powers the visual and interactive aspects of the Cryptoverse, supporting high-resolution environments and realistic animations.
- **Blockchain Integration:** Utilizes blockchain technology for asset ownership, governance, and transactions.
- **Smart Contracts:** Manage LAND sales, in-game assets, and decentralized governance.

2. LAND and Virtual Space

- **LAND as NFTs:** Cryptoverse features virtual real estate divided into NFT-based LAND parcels.
- **Dynamic World Structure:** The metaverse is divided into **eight zones**, each with unique aesthetics and functions.
- **Modular Building System:** Allows users to create, modify, and customize digital spaces without requiring advanced programming skills.

3. User Interaction & Economy

- **Avatar System:** Users can create and customize avatars for interaction.
- **Marketplace & Economy:** Features a digital economy where users can buy, sell, and trade assets using the native $CGG token.
- **Prana System:** A user engagement metric that rewards participation and activity.

3.Setup Instruction

To set up and access the Cryptoverse, follow these steps:

1. Visit the Official Cryptoverse Website:

- Navigate to the Cryptoverse official website

2. Create an Account:

- Click on the "Sign Up" or "Register" button.
- Provide the required information, such as your email address, username, and password.
- Verify your account via the confirmation email sent to your registered email address.

4.Folder structure

The **Cryptoverse folder structure** will depend on whether you are a **user** exploring the metaverse or a **developer** working on modifications or integrations. Below is a general breakdown of how a Cryptoverse directory might be structured based on typical metaverse projects built with **Unreal Engine 5** and blockchain components.

1. User Installation Folder Structure

If you install the Cryptoverse client, your directory may look like this:

```
bash
CopyEdit
Cryptoverse/
├── Binaries/               # Executable files for
launching the game
├── Config/              # Game configuration files
├── Content/                # 3D models, textures,
sounds, UI assets
├── Logs/                # Error logs and debugging
information
├── Plugins/               # Additional features or
extensions
```

```
├── Saved/                  # User settings and local
saved files
├── Cryptoverse.exe            # Main executable for
launching the game
├── README.md                # Documentation or setup
guide
```

---

2. Developer Folder Structure (Unreal Engine 5 Project)

If you are developing or modifying Cryptoverse using **Unreal Engine 5**, the directory structure might look like this:

```
graphql
CopyEdit
Cryptoverse-Project/
├── Source/                 # C++ source code for the
game
├── Plugins/                  # Custom Unreal Engine
plugins
├── Content/                # 3D assets, animations,
and UI elements
│   ├── Maps/             # Different world zones and
maps
│    ├── Characters/         # Player and NPC models
│    ├── UI/                 # User interface assets
├── Config/              # Unreal Engine configuration
files
├── Blockchain/          # Smart contract integrations
│    ├── Ethereum/          # Ethereum-based contracts
(e.g., LAND NFTs)
│     ├── Polygon/               # Layer-2 blockchain
contracts
├── Marketplace/              # NFT marketplace and
transaction scripts
├── Logs/                  # Log files for debugging
├── Cryptoverse.uproject     # Unreal Engine project
file
├── README.md                   # Documentation for
developers
```

---

3. Blockchain Integration Folder (for Smart Contracts & Transactions)

If you're working on **blockchain-related components**, a typical structure might be:

```
bash
CopyEdit
Blockchain/
│── Contracts/                # Smart contracts written
in Solidity
│   │── LAND.sol              # NFT contract for
virtual real estate
│   │── Marketplace.sol       # NFT trading contract
│── Scripts/                  # Web3 integration
scripts (e.g., Node.js, Python)
│── ABI/                      # Compiled contract ABI
files for frontend integration
│── Tests/                    # Unit tests for
blockchain contracts
│── README.md                 # Documentation for
blockchain integration
```

---

4. Web-Based Dashboard or Marketplace

If Cryptoverse includes a web-based **LAND marketplace or user dashboard**, the structure could be:

```
php
CopyEdit
Web-Dashboard/
│── public/                     # Static assets (images,
icons, etc.)
│── src/                      # Source code for frontend
│   │── components/           # React/Vue components
│   │── pages/                # Web pages
│   │── utils/                # Utility functions (Web3,
API calls)
│── smart-contracts/          # Blockchain interactions
│── package.json             # Dependencies for frontend
│── README.md                 # Instructions for setting
up the dashboard
```

5.Running the Application

Depending on whether you're a **user** or a **developer**, the steps for running Cryptoverse may vary. Below are the instructions for both cases.

1. Running Cryptoverse as a User (Installed Client)

If you have installed the Cryptoverse **game client**, follow these steps:

Step 1: Launch the Application

- Navigate to the Cryptoverse installation directory:

  ```
  makefile
  CopyEdit
  C:\Program Files\Cryptoverse\
  ```

- Double-click the **`Cryptoverse.exe`** file to launch the application.

Step 2: Log In

- Enter your registered **username** and **password**.
- If using **Web3 authentication**, connect your **MetaMask wallet** or compatible blockchain wallet.

Step 3: Explore the Metaverse

- Customize your **avatar**.
- Navigate through different **zones**.
- Buy, sell, or rent **LAND** (if applicable).

---

2. Running Cryptoverse as a Developer (Unreal Engine 5 Project)

If you are a developer working with **Unreal Engine 5**, follow these steps:

Step 1: Clone or Open the Project

- Open a terminal and navigate to your development folder:

  ```
  bash
  ```

```
CopyEdit
cd path/to/Cryptoverse-Project
```

- If the project is stored in a **Git repository**, clone it:

```
bash
CopyEdit
git clone https://github.com/Cryptoverse/cryptoverse-project.git
```

- Open **Unreal Engine 5** and select **Cryptoverse.uproject**.

Step 2: Compile the Project

- If using **C++**, open the project in **Visual Studio**.
- Build the project by running:

```
CopyEdit
UE5Editor.exe Cryptoverse.uproject
```

Step 3: Run the Application in Editor Mode

- Click **"Play"** inside Unreal Engine to launch the Cryptoverse.
- Adjust the **settings** to optimize graphics and performance.

Step 4: Package and Run the Standalone Version

- In Unreal Engine, go to:

```
mathematica
CopyEdit
File → Package Project → Windows/macOS/Linux
```

- After packaging, navigate to:

```
swift
CopyEdit
Cryptoverse-Project/Saved/Build/
```

- Run the generated **Cryptoverse.exe** file.

---

6. Component Documentation

1. Core Infrastructure

- **Unreal Engine 5:** High-fidelity graphics & physics.
- **Blockchain Integration:** Smart contracts for NFTs & LAND.
- **Networking:** Multiplayer & decentralized interactions.

2. Virtual World (Metaverse)

- **LAND (NFTs):** Buy, sell, or develop virtual real estate.
- **Zoning System:** 8 unique zones for different activities.
- **Customization:** User-created content & environments.

3. User Interaction & Economy

- **Avatar System:** Customizable characters.
- **Marketplace:** Trade NFTs & digital assets with $CGG tokens.
- **Prana System:** Engagement-based rewards.

4. Web3 & Blockchain Layer

- **Ethereum/Polygon Smart Contracts:** NFT-based LAND ownership.
- **MetaMask & WalletConnect:** Web3 authentication.
- **Decentralized Governance (DAO):** Community-driven decisions.

5. Development Tools & API

- **Unreal Engine Plugins:** For extensibility & modding.
- **Web3 API:** For interacting with blockchain assets.
- **Developer SDK:** Tools for asset creation & scripting.

7. State Management

1. Overview

Cryptoverse uses a **hybrid state management system** combining **client-side**, **server-side**, and **blockchain-based states** for seamless metaverse interactions.

---

2. Client-Side State (Frontend)

- Managed using **React (for UI)** and **Unreal Engine 5 (for 3D interactions)**.
- **State Handling:**
  - **React Context API / Redux** (for UI components).
  - **UE5 Game Instance & Save System** (for real-time session states).
- **Key States:**
  - Avatar customization
  - UI elements (inventory, chat, marketplace)

---

3. Server-Side State (Backend)

- Uses **Node.js, Express, and WebSockets** for real-time updates.
- **State Handling:**
  - **Database (PostgreSQL, Firebase, or MongoDB)** for persistent data.
  - **Redis**

8. User Interface

The **Cryptoverse UI** is designed for an immersive and seamless metaverse experience, integrating **3D navigation, Web3 features, and user customization**.

---

1. Main UI Components

🔹 Dashboard (Home Screen)

- **Profile Overview** – Avatar, wallet balance, and settings.

- **Quick Access Menu** – LAND ownership, marketplace, social features.

◈ *Navigation & Interaction*

- **Mini-Map & World Zones** – Helps users explore different areas.
- **Teleportation & Fast Travel** – Move quickly within Cryptoverse.
- **Chat & Social Features** – Text & voice chat, friend list, and groups.

10.**Styling**

Cryptoverse follows a **futuristic, immersive, and user-friendly** design language optimized for a **metaverse experience**.

---

1. Visual & UI Styling

◈ **Futuristic & Cyberpunk Aesthetic** – Neon lights, holograms, and sleek UI elements.
◈ **Minimalist & Intuitive UX** – Easy-to-navigate menus and smooth transitions.
◈ **Dark & Neon Color Palette** – High contrast for immersive metaverse visuals.

---

2. UI Styling Technologies

◈ For Web (React-based UI):

- **CSS-in-JS (Styled Components, TailwindCSS).**
- **Glassmorphism & Neomorphism** effects for futuristic UI.

◈ For 3D UI (Unreal Engine 5):

- **Blueprint UMG Widgets** for in-game UI elements.
- **HUD Components (Icons, Chat, Menus)** optimized for VR/AR.

11. Testing (Quick Overview)

Cryptoverse uses a **multi-layered testing approach** to ensure stability, security, and performance across **gameplay, Web3 interactions, and UI components**.

---

1. Testing Types & Methodologies

📏 *Unit Testing (Code-Level Verification)*

- **Backend:** Jest, Mocha (Node.js services, API responses).
- **Smart Contracts:** Hardhat, Truffle (Blockchain transactions & NFT logic).
- **Game Engine:** Unreal Engine Unit Tests for gameplay mechanics.

📏 *Integration Testing (Feature-Level Testing)*

- **Web3 Transactions:** Simulated transactions using Ganache & Hardhat.
- **Database & Server:** Ensuring PostgreSQL/MongoDB syncs with in-game data.
- **Gameplay Systems:** Multiplayer interactions, in-game marketplace, AI behavior.

📏 *UI/UX Testing (User Experience & Responsiveness)*

- **Frontend:** Cypress, Selenium (Automated UI testing for React-based dashboard).
- **3D UI:** Unreal UMG interaction tests (HUD, menus, avatar customization).
- **VR/AR Compatibility:** Testing UI responsiveness in VR environments.