TED UNIVERSITY

Faculty of Engineering

Departments of Computer & Software Engineering

Project Name: **DiscoVeR**

High-Level Design Report

Team Members:
CMPE - Levent Bektemur 13759162240
CMPE - Dilge Nisa Üstüntaş 13117245554
SENG - Süzanna Nebioğlu 25123817076
Supervisor:
Ulaş Güleç
Jury Members:
Tansel Dökeroğlu
Emin Kuğu

Link to the web page of the project:
https://humanfromnowhere.github.io/discover.github.io/

Table of Contents

1. Introduction

1.1 Purpose of the System

The Discoverer system's goal is to offer a web-based virtual reality (VR) touring platform that lets users turn their own models of 3D environments into interactive VR experiences that can be accessed through a web browser. The system is intended for content producers who wish to digitally promote physical locations, such as hotel owners, real estate brokers, or travel agencies. Discoverer seeks to integrate reliable 3D rendering with web accessibility by incorporating Unity WebGL into a contemporary web framework.

1.2 Design Goals

The primary design goals of DiscoVeR are:

- **Web-based access:** Use WebGL to embed the Unity engine so that the system runs in a regular web browser.
- **Ease of use:** Offer a user-friendly model uploading and management interface.
- **Performance:** Use the Unity engine to produce steady frame rates and high-devotion virtual reality experiences.
- **Scalability:** Allow for growing user and model counts.
- **Security and Privacy**: Protect user accounts and proprietary 3D assets.

1.3 Definitions, Acronyms, and Abbreviations

- **VR:** Virtual Reality
- **WebGL:** Web Graphics Library
- **Unity:** A cross-platform game engine used for creating 3D content.
- **Node.js:** A JavaScript runtime environment for server-side development.
- **FBX/OBJ:** 3D model file formats

1.4 Overview

The Discover system's high-level design is presented in this document. To ensure reliable web management and rendering, this document outlines modifications to the Unity-based architecture integrated into the Node.js application structure.

2. Current Software Architecture

The Discover platform is intended to function as an independent web-based system. The architecture exports projects to WebGL so they may run directly in browsers, utilizing Unity's potent 3D physics and rendering capabilities. This eliminates the requirement for lower-level library implementations (like raw Three.js), preserving web compatibility while enabling more intricate interaction logic and visual fidelity.

3. Proposed Software Architecture

3.1 Overview

The proposed architecture follows a hybrid client-server model:

1. **Web Application and Admin Panel:** Developed using Node.js. This layer handles the user interface for registration, login, dashboard, and file management.
2. **3D Visualization Core: Developed using Unity.** The VR touring environment is built in Unity and exported as a WebGL build.
3. **Integration:** Unity WebGL builds are embedded directly into Node.js application views. When a user requests to view a tour, the Node.js backend serves a specific HTML wrapper that initializes a Unity instance.

3.2 Subsystem Decomposition

The system is decomposed into the following major subsystems:

- **User Management Subsystem (Node.js):** Handles user registration, authentication, authorization, and password recovery.
- **Model Upload and Management Subsystem (Node.js):** Manages the upload, validation, storage, listing and deletion of 3D models through the web admin panel.
- **VR Rendering Subsystem (Unity WebGL):** Responsible for loading processed models and rendering them in an interactive VR environment. This subsystem runs inside the browser canvas, powered by the Unity engine.
- **Frontend User Interface Subsystem (Node.js/HTML5):** Provides a DOM-based UI for dashboard and upload pages, and acts as a container for the Unity WebGL canvas.

3.3 Hardware/Software Mapping

Client side: Operates on end-user devices with browsers that support WebGL. The browser runs Unity WebGL assembly (WASM) for the 3D experience and Node. js-generated scenes for the user interface.

Server Side: Node.js programs operate on the server. allows file storage, manages authentication, delivers Unity build files, and provides web application assets.

3.4 Persistent Data Management

Persistent data includes user account information and metadata about uploaded models.

Database: A database connected to the Node.js backend stores user credentials and model metadata.

File Storage: 3D model files and the compiled Unity WebGL build assets are stored in the server's file system or cloud storage.

3.5 Access Control and Security

Access control is enforced by the Node.js backend.

- **Authentication:** Only authenticated users can access the admin panel to upload or delete models.
- **Unity Security:** The Unity instance communicates with the backend via secure API calls to fetch model data, ensuring that unauthenticated users cannot access restricted assets directly.

3.6 Global Software Control

Global control flow is event-driven.

1. The user logs in via the Node.js web interface.
2. The user uploads a model; the backend validates and stores it.
3. When "View" is clicked, the Node.js server routes the user to a page embedding the Unity WebGL player.
4. The Unity player initializes and requests the specific model data to render the scene.

3.7 Boundary Conditions

WebGL Support: The Unity Loader's fallback message will appear in browsers that do not support WebGL.

Performance: Unity WebGL load times may be affected by large models; loading times and optimization warnings will be applied by the system.

4. Subsystem Services

Authentication Service (Node.js): Register, log in, log out.

Model Management Service (Node.js): Upload model, list models, delete model.

Visualization Service (Unity): Render 3D environment, handle VR camera controls, process user input within the 3D space.

Integration Service: Bridges the communication between the DOM which is known as web UI and the Unity Canvas.

5. Glossary

Content Creator: Authenticated user who uploads 3D models.

Viewer: End-user who accesses and explores VR scenes.

Unity WebGL: A build option in Unity that allows content to be published as JavaScript/WebAssembly programs.

6.References:

IEEE. (n.d.). *IEEE code of ethics*. Retrieved December 24, 2025, from https://www.ieee.org/about/corporate/governance/p7-8.html

Jones, B., & Holkner, A. (Eds.). (2024, February 13). *WebXR device API*. W3C. https://www.w3.org/TR/webxr/

Khronos Group. (n.d.). *glTF 2.0 specification*. Retrieved December 24, 2025, from https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html

OpenJS Foundation. (n.d.). *Node.js documentation*. Retrieved December 24, 2025, from https://nodejs.org/en/docs/

Unity Technologies. (n.d.). *Unity manual: WebGL*. Retrieved December 24, 2025, from https://docs.unity3d.com/Manual/webgl.html