# CaveMap

**Group 6 - Haitian Zhu, Sola Akindele, Dale James Joseph, Khizer Khan**

## Overview

The CaveMap project is designed as a distributed hardware-software system that delivers real-time underground mapping and navigation in GPS-denied environments. At the core of the design, the split-client architecture separates intensive computational processing from the lightweight user viewing interfaces. This architectural decision ensures performance, reliability, and usability in extreme subterranean conditions. The system integrates a wearable mask equipped with LiDAR sensors, cameras, and an Augmented Reality (AR) Heads-up Display to capture spatial and visual data directly from the user's environment. Because real-time 3D mapping and AI-based mineral detection require significant computational power, all heavy processing is propagated to a main client hub, which handles data-heavy tasks and broadcasts results back to the user. The primary purpose is to create a reliable technological platform capable of generating live 3D cave maps, detecting environmental hazards, and assisting with navigation.

## Functional and Performance Requirements

The technical specifications for the CaveMap ecosystem focus on engineering a high-reliability system that ensures personnel safety in high-risk environments through sensor fusion. Functional requirements specify core capabilities including real-time LiDAR processing, AR-assisted pathfinding, and AI-driven geological feature tagging using spatial octrees for efficient 3D indexing. To meet life-safety standards, the system is mandated to maintain a refresh rate of 10Hz and a spatial accuracy of 5cm, ensuring the digital representation reflects physical reality. Performance standards require a total system latency under 150ms to maintain spatial orientation during active movement.

## Proposed System Architecture

The proposed system architecture adopts a hybrid approach, combining the Model-View-Controller (MVC) design pattern with a Three-Tier Client-Server model. The Frontend (View) acts as the Client, providing the user-facing interface for interaction with 3D wireframes and safety alerts across wearable AR headsets and mobile sub-clients. The Backend (Controller and Model) operates on the local Main Client server, managing complex spatial logic and the 3D SLAM engine. This MVC structure allows for the separation of concerns, where the AI detection models can be updated independently of the AR rendering interface. The Client-Server model enables a distributed workload, offloading computation-intensive SLAM processing to a ruggedized local server while keeping the wearable headset lightweight and thermally efficient for extended subterranean usage.

## Subsystem Decomposition and Object Design

The CaveMap system is organized into six cooperating subsystems arranged around an event-driven core. The User Interaction Subsystem facilitates the primary interface for mission initialization and mode selection, while the Mapping Subsystem serves as the computational heart, utilizing the SLAM Engine and OctreeBuilder to maintain the cave geometry. The Data Integration Subsystem manages the hardware interface with the LiDARController to synchronize raw point-cloud data. The Data Management Subsystem governs the MapDatabase for persistent storage, and the Visualization Subsystem utilizes a SensorCache to buffer incoming data for the ProcessPipeline. Finally, the Security Subsystem enforces role-based access control (RBAC) and AES-256 encryption across all data boundaries to protect sensitive geological findings and user privacy.

## Hardware and Software Mapping

The hardware mapping is designed around a localized mesh network to ensure functionality where cellular and satellite signals are unavailable. The Wearable Sensor Unit serves as the primary client, capturing raw data via LiDAR arrays and thermal cameras. The Main Client acts as the local high-performance server, handling all business logic and simulation processes locally to preserve real-time responsiveness. Remote Viewers, such as ruggedized tablets and laptops, serve as sub-clients for researchers and rescue teams, interfacing with the Main Client to provide live-streaming and historical map browsing. Communication is facilitated through high-speed wired links for sensor-to-server data and a local WiFi mesh network for server-to-sub-client broadcasts, ensuring a unified "team view" of the exploration zone.

## Global Control and Boundary Conditions

Global software control is organized around an event-driven, pipeline-based controller pattern utilizing dedicated SensorIngest, Processing, and Output threads. A central SystemController owns lifecycle concerns, including startup self-tests of hardware, integrity checks of persistent files, and graceful shutdown protocols. Boundary conditions are treated as first-class concerns; on startup, the system performs a hardware self-test with per-device reporting and a cold-boot recovery mandate that ensures hazard sensors are active within 15 seconds. In the event of an abnormal shutdown or power loss, a RecoveryManager replays append-only logs to rebuild the last consistent octree state, ensuring that mission-critical data remains protected even in the harshest environmental conditions.

## User Interface and Human Factors

The human-in-the-loop experience is optimized through a high-contrast AR HUD design that minimizes cognitive load in high-stress, low-visibility conditions. The interface provides a 3D wireframe overlay for navigation, a proactive hazard alert system, and AR breadcrumbs to guide explorers back to safety. For remote personnel, the Sub-Client Dashboard offers a rotatable 3D map and a telemetry feed of atmospheric conditions.