CSC 431

<Call.US>

System Architecture Specification (SAS)

<14>

Kai Pettini	Technical Architect
Ximena Chen	Diagram Designer
Fey Adenrele	Technical Architect
<Member Name>	<Role>

Version History

Version	Date	Author(s)	Change Comments
1.1	04.20	Kai Pettini	App functionality focused on Meta's Quest and Apple's Vision Pro

Table of Contents

31. System Analysis

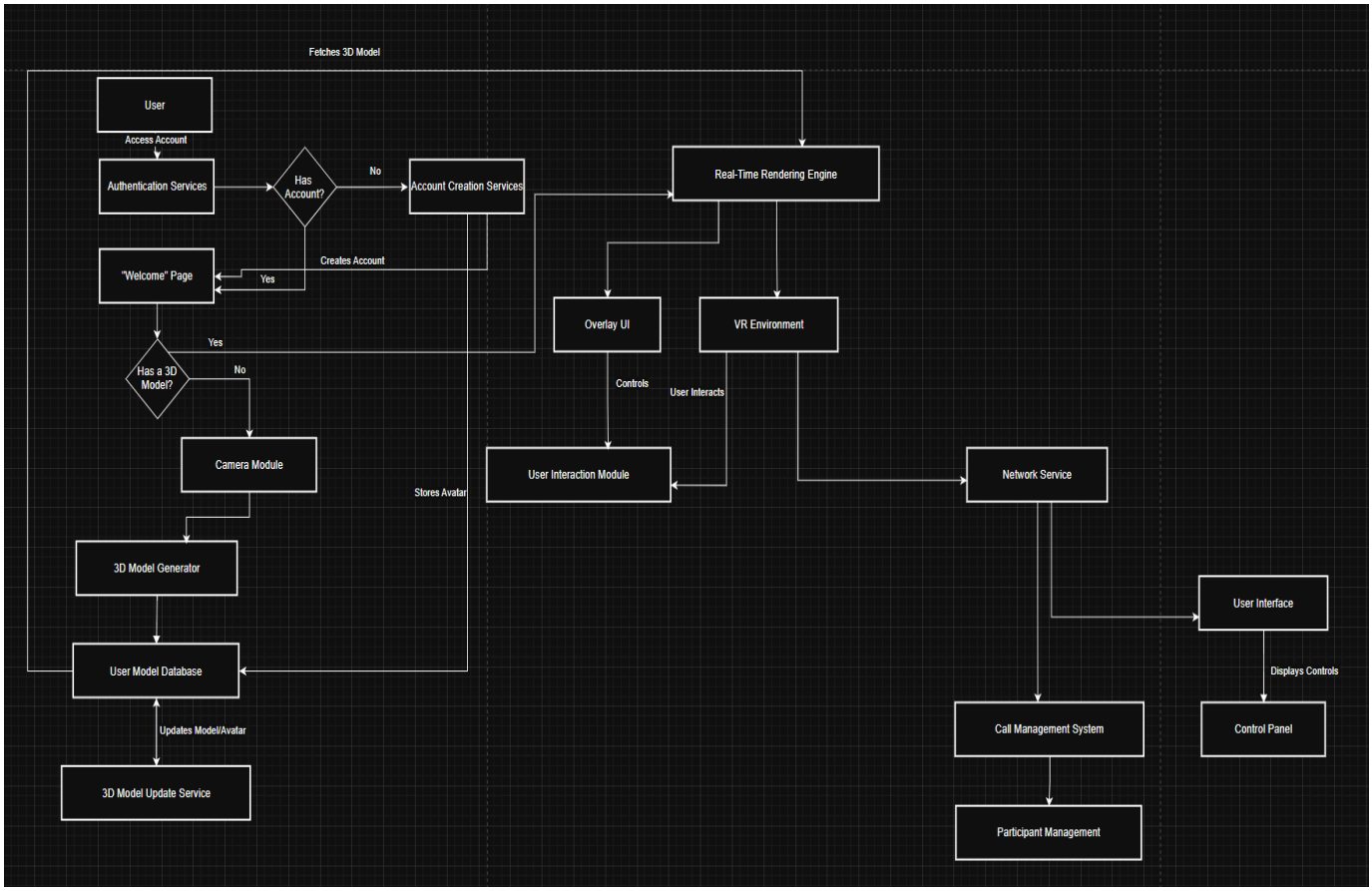
31.1 System Overview

Call.US is an in-real time video-calling application, mainly focused on mobile devices, that uses that device's sensors to construct a 3D model of the user's body. The user will then be able to use this generated model in order to call their other users in a 3D world. A strong support for Virtual Reality Headsets, like Meta's Quest, will be given to this app, as the main objective is to provide users with the experience of meeting their friends or family, providing a close and personal scenario in the process.

Call.US will also integrate video and audio streaming, VR rendering and motion tracking, and a secure database that will hold the information of the users.

Call.US will also use a mix of Client-Server, and Layered architectures. Due to the expected large number of users and necessity to connect to a central server, the Client-Server architecture fits this project really well. Furthermore, due to the large number of functions this application will provide, the Layered architecture would provide a great foundation for the application, as it will help to structure the project in a more efficient and practical way.

31.2 System Diagram



31.3 Actor Identification

User: Represents the main user of the application. They are the ones to start and end calls, interact with the UI, manage their account, etc. It mainly interacts with the other users, sharing their 3D model and having video and audio provided by the system.

Moderator: Responsible for administering other users, scanning for possible breaches of the terms of services, applying restrictions and bans to users depending on the gravity of their actions.

Customer Service: Responsible for answering complaints, responding to questions, giving feedback to possible app problems.

VR Sensors: These are responsible for tracking movement and also providing video and audio for the user.

Rendering Engine: Displays the 3D body models in real time. This will get their info based on the user accessing it, and update this model movements based on the interactions and movement outputs provided by the VR Sensors.

Authenticator: Manages sign-in, registration, profile data. Stores the information provided in a database when an account is created.

Communication Server: Manages communication between users during calls. Does this by managing the video and audio in data packets. Also synchronizes environments in which more users are present.

31.4 Design Rationale

1.4.1 Architectural Style

Hybrid Architecture (Client-Server + Layered Architecture)

The hybrid architectural style of the Call.US system mainly integrates:

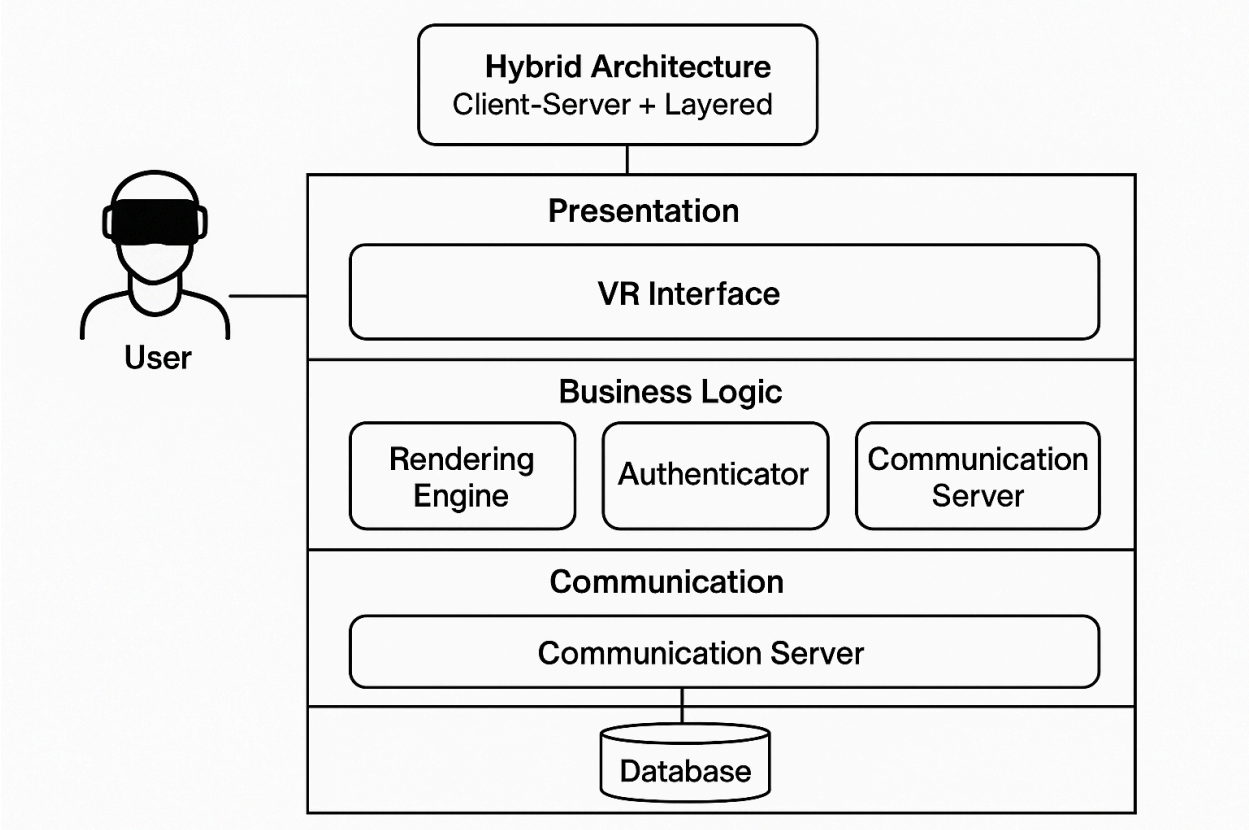
- ***Client-Server Architecture:*** *Crucial for centralizing 3D model data, facilitating real-time video and audio streaming, and controlling user communication. It makes centralized control and scalability possible, which is essential for upholding user moderation rules and preserving performance.*
- ***Layered Architecture:*** *To provide more flexibility and maintainability, the system is organized into logical layers: Presentation, Business Logic, Communication, and Data. To enhance modularity and testing, for example, the rendering engine (Presentation Layer) functions separately from the database and authentication (Data Layer).*

  In order to guarantee scalability, performance, and separation of concerns all crucial for a real-time VR communication platform this hybrid method was selected.

1.4.2 Design Pattern(s)

The development of Call.US has been organized and supported by a number of design patterns:

- *In the frontend, particularly in mobile apps and virtual reality interfaces,* ***Model-View-Controller (MVC)*** *is used to divide the user interface (View), application logic (Controller), and data (Model). Clean code, simpler testing, and the ability to change the interface without changing the logic are all facilitated by this.*
- ***Observer Pattern:*** *Perfect for managing 3D model synchronization and real-time changes in audio and video streams. For instance, the rendering engine updates the avatar's position in real time based on changes in motion data from the VR sensors.*
- *The Authenticator and Communication Server components use the singleton pattern to guarantee a single point of access for secure communication handling and session control.*
- ***Facade Pattern:*** *Enables developers and external systems to communicate with a single API by offering a simplified interface to intricate subsystems like VR rendering and sensor integration.*



1.4.3 Framework

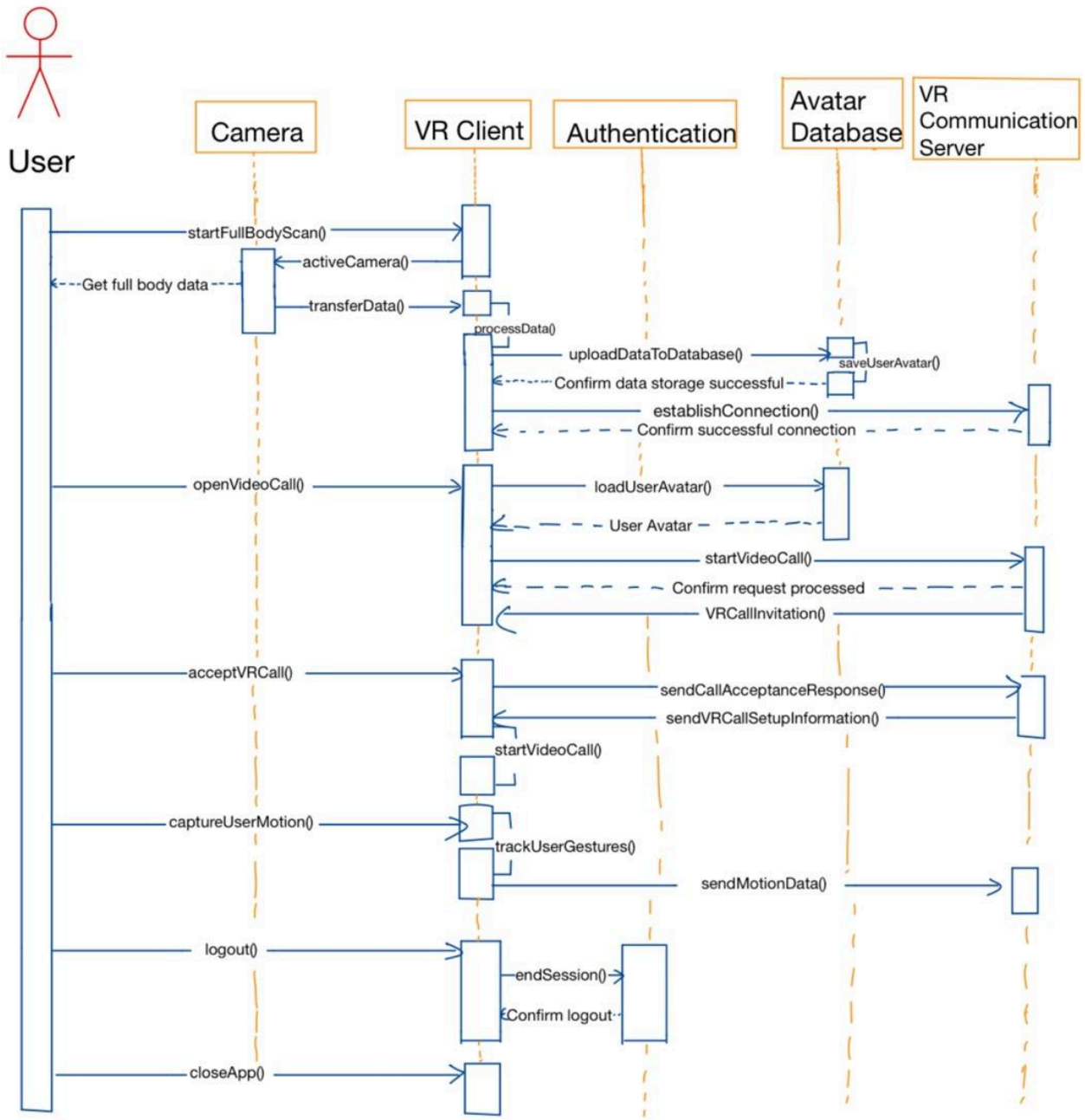
Unity 3D with VR SDKs (Meta Quest SDK, Apple Vision Pro SDK)
Backend: Node.js + Express.js
Database: Firebase Realtime Database

- Unity 3D was selected as the main development environment for the VR components due to its powerful rendering engine, cross-platform support, and seamless integration with Meta Quest and Apple Vision Pro SDKs. These SDKs allow access to headset sensors, spatial audio, and gesture recognition, crucial for delivering an immersive 3D calling experience.
- Node.js + Express.js is used for the backend server due to its non-blocking I/O, making it ideal for handling real-time video/audio streaming and concurrent user connections.
- Firebase is chosen for its real-time synchronization, ease of use, and secure authentication capabilities, which are beneficial for storing user profiles and session data.

- Rationale: This framework combination was selected to balance performance, developer productivity, and real-time responsiveness, all of which are critical in a VR-driven communication platform. Unity offers immersive 3D environments, while Firebase ensures low-latency data syncing, and Node.js handles the backend logic efficiently.

32. Functional Design

32.1 VR Video Call System Sequence Diagram



33. Structural Design