StratoSplit

Software Design Document

February 15, 2025

Client:

General Dynamics Mission Systems

Mentors:

Brian Donnelly, Savannah Chappus

Team Members:

Sam Cain

Nolan Newman

Dallon Jarman

Elliot Hull

Revision 1.2

Table of Contents

Introduction

Effective communication is paramount in defense, public safety, and intelligence operations. In high-stakes environments, rapid and reliable information exchange is critical for mission success. From coordinating search-and-rescue efforts to relaying real-time intelligence, organizations depend on secure and efficient communication systems. However, existing technologies often struggle to meet modern demands.

General Dynamics Mission Systems (GDMS), a leading defense contractor, specializes in mission-critical products and systems. Their notable projects include Rescue 21, a Coast Guard distress location system, and the next-generation Global Positioning System (GPS). GDMS has tasked our team with creating a solution that enhances communication efficiency.

Current communication with radio modems relies on cumbersome web interfaces. These systems lack intuitive design, making real-time data transmission inefficient and limiting the ability to manage audio attachments seamlessly. Additionally, performance issues arise under varying network conditions, further complicating effective communication between operators and field personnel.

The StratoSplit project aims to address the challenge of inefficient communication with radio modems on mobile devices. Inspired by General Dynamics Mission Systems' innovative solutions, our team is developing a Node.js web application, StratoSplit, to simulate audio generation and transmission to a dashboard for real-time monitoring.

The core modules of our solution consist primarily of an audio generator, a web application, unit testing, and security. The audio generator is a virtual machine hosted on Amazon Web Services GovCloud (AWS GovCloud) responsible for generating Real-time Transport Protocol (RTP) audio packets and sending them to our web application on another virtual machine on AWS GovCloud via multicast addresses. This web application consists of the following components: a database, a user interface, and server side spatial audio processing.

StratoSplit aims to revolutionize communication in defense, public safety, and intelligence communities by providing a secure, efficient, and user-friendly web application for simulated audio generation and transmission. Our team is committed to delivering a high-quality solution that meets the needs of GDMS and its clients.

Implementation Overview

The vision for StratoSplit is to develop a highly efficient and scalable audio stream generator and dashboard that enables real-time collection, processing, and playback of radio signals through the cloud. This platform will allow users to seamlessly access radio communications from a centralized system, with the audio streams being securely transmitted to their local machines. By leveraging cloud-based infrastructure, we aim to provide a high-performance, resilient, and user-friendly advanced audio management experience.

The key features are as follows, simulated audio generation, real-time audio transmission, a dashboard for monitoring, and a user-friendly interface. The functional requirements are as follows, the application shall simulate audio generation, the application shall transmit audio to a dashboard in real time, and the application shall provide a user-friendly interface. Additionally, the non-functional requirements include ensuring high availability, optimal performance under varying network conditions, and adherence to industry-standard security protocols.

These features will be integrated with the following frameworks, services, and languages:

- MongoDB: a NoSQL database solution which we will use to store user credentials and configurations. This was chosen based on the client's request for a NoSQL database.
- Amazon Web Services GovCloud (AWS GovCloud): A cloud computing platform designed for U.S. government agencies and contractors.This was chosen based on the client's providing our team access to its resources.
- Node JS: A JavaScript runtime that will serve as the backbone of our application, handling server-side logic and enabling real-time communication between clients and the audio transmission system. This was chosen because of the built in support for audio processing, web sockets, and high potential for parallel processing.
- Python3: A programming language that will be used for signal processing and audio data manipulation within the audio generator. Python's extensive libraries for scientific computing and networking make it well-suited for these tasks.
- Real-time Transport Protocol (RTP): A standard protocol for delivering audio and video over IP networks. We chose RTP specifically due to its compatibility within multicast systems and ease of integration within the client's current system.
- Docker: A containerization platform that will allow us to package the application and its dependencies in a consistent environment. This was chosen after internal testing concluded it was the best option for our needs.

- Ubuntu 24.10: The linux distribution we will use as the operating system for our deployment environment. This was chosen because of the ability to modify kernel code to enable Internet Group Management Protocol Version 2 (IGMPv2) in the AWS GovCloud environment for multicast transmissions.
- Jest: A JavaScript testing framework that can be used to ensure the reliability and correctness of Node.js applications. Jest was picked due to its built-in support for unit, integration, and snapshot testing as well as code coverage measurements.

By integrating these features into a seamless, cloud-powered solution, StratoSplit will deliver a secure, reliable, and immersive audio streaming experience tailored to users who require advanced, real-time communication capabilities. With a strong foundation in cloud computing and a focus on future scalability, our platform will continue evolving to meet the needs of its users, offering an unprecedented level of control and flexibility in managing radio communications. With this flexible and modular design, we pave the way for additional features such as mobile support, sophisticated audio processing, and AI-driven enhancements.

Architectural Overview

To achieve seamless real-time communication, StratoSplit is built on a secure and scalable architecture that integrates Amazon Web Services GovCloud hosted virtual machines, Docker containerization, and a robust MongoDB system for user authentication and data management. The system is designed to handle varying workloads while maintaining optimal performance, incorporating logging mechanisms, automated testing suites, and a modular deployment approach for seamless debugging, upgrades, and reliability.

The hosting infrastructure relies on AWS GovCloud, providing a secure and compliant environment for deployment. StratoSplit uses Amazon EC2 virtual machines (VMs) to host key components. One EC2 instance runs the console web application, a Node.js-based system deployed within a Docker container. This web app manages user authentication, processes requests, and facilitates real-time interaction. A separate EC2 instance hosts the audio stream generator, which simulates radio communications and transmits audio streams for processing. Users interact with the system via an operator workstation, accessing the web application through a browser. Once authenticated, users can manage and monitor active audio streams, which are transmitted from the audio stream generator to the web server for real-time playback.

At the heart of the system is the web server, which ensures low-latency audio transmission and real-time processing. Built using Node.js, the server handles audio data routing, authentication, and access control. Audio streams transmitted via Real-time Transport Protocol (RTP) are received, processed, and distributed to connected users. The web server also manages interactions with the MongoDB database, verifying user credentials and retrieving configuration data. To improve deployment efficiency and ensure consistent performance, the entire web server is containerized using Docker, allowing seamless scaling and fault isolation. This microservices-style approach ensures that each system component operates independently while maintaining efficient communication.

For authentication and data management, StratoSplit employs MongoDB, a NoSQL database that securely stores user credentials, authentication keys, and audio stream configurations. Hosted on a dedicated external server, MongoDB provides scalable and flexible data storage, allowing the system to adapt to evolving requirements without rigid schema constraints. Role-based access control is enforced, ensuring that operators can monitor and control streams while administrators manage permissions and configure access to specific radio channels.

Real-time audio transmission is achieved through RTP, enabling the audio stream generator to simulate live radio communication. Audio packets are transmitted over the network, processed by the web server, and forwarded to authorized users. To optimize bandwidth usage while preserving audio quality, the system uses G.711 mu-law encoding.

The user interface is designed for clarity and usability, offering an intuitive browser-based dashboard where users can manage and monitor audio streams. Integrated with Node.js, the dashboard provides real-time updates using WebSockets, ensuring minimal latency when interacting with the system. Users can adjust volume levels, mute individual channels, and group streams for easier management. Administrators have additional controls, such as managing user roles, configuring access permissions, and creating or deleting accounts. Audio output is handled through speakers connected to the operator workstation, ensuring a seamless listening experience.

To maintain system reliability and performance, StratoSplit integrates Jest for automated testing, covering unit, integration, and performance tests. These tests verify that audio interactions remain consistent. Additionally, the web server and database feature comprehensive logging mechanisms,allowing for real-time monitoring, debugging, and security auditing. These logs ensure compliance with operational standards and provide insights into system performance.

StratoSplit's architecture is designed to provide a secure, scalable, and efficient solution for real-time audio communication in mission-critical environments. By leveraging AWS-hosted virtual machines, containerized deployment, and a robust web server, the system ensures seamless audio transmission while maintaining high availability and fault tolerance. The integration of MongoDB for authentication, RTP for real-time streaming, and Jest for automated testing within a Node.js environment reinforces reliability, security and performance. StratoSplit is built to handle evolving operational demands while ensuring low-latency, high-quality communication.

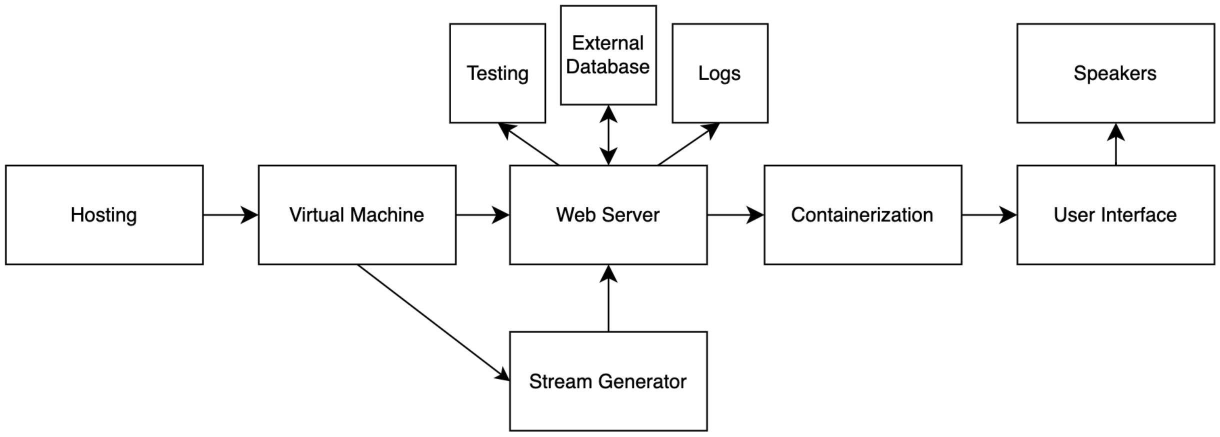


Figure 1: An overview of the systems infrastructure, showing how components interact within a cloud-hosted environment.

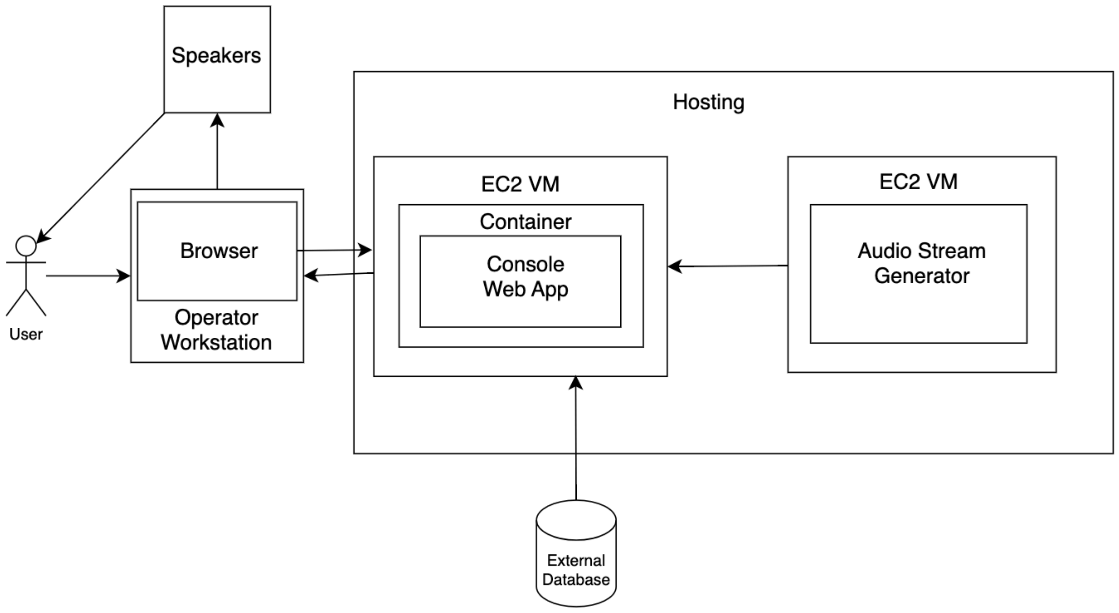


Figure 2: A a detailed view of user interaction with the system, demonstrating how the operator workstation, web application, and audio output devices are integrated to deliver an enhanced audio experience

Module and Interface Descriptions

The Audio Stream Dashboard is a web-based application that is designed to provide a secure, efficient, audio stream management system allowing the Coast Guard to quickly and effectively receive one-way communication from ships out at sea as well as adhere to General Dynamics requirements. This comprehensive system enables users to monitor and control multiple audio channels through an intuitive spatial audio interface. This section will talk about the Audio Generator, and the Web Application, with the application broken down into five main points, the database, user interface, spatial audio, testing, and security.

Audio Generator

The Audio Stream Generator is responsible for producing real-time audio streams and delivering them over RTP multicast. It runs on an EC2 VM and responds to requests for audio generation, dynamically creating and transmitting RTP packets to a multicast group.

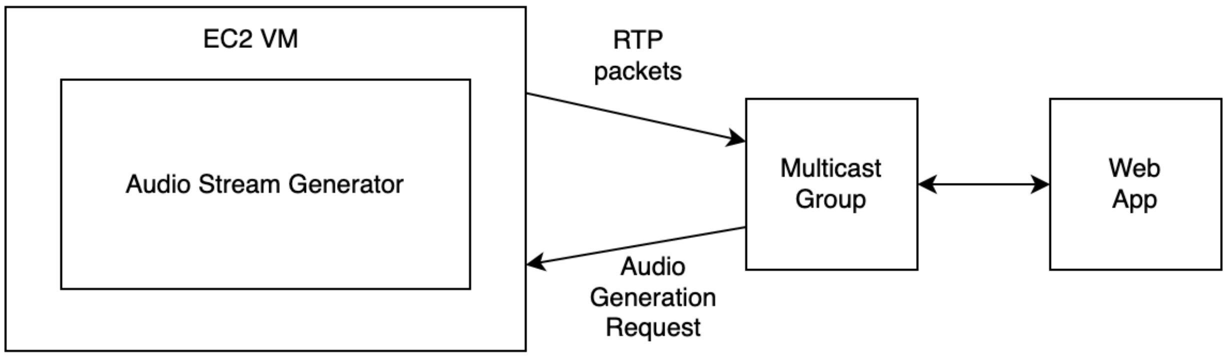


Figure 3: Audio Generator Component Diagram

Endpoints:

- Audio Generation Request: This endpoint allows an authorized administrator to initiate an audio stream. Upon success, the system returns a unique stream ID and the multicast address where the audio is transmitted. Unauthorized users will receive an access denied response.
- Stream Information Lookup: Returns the status of an ongoing preselected stream. Users can only request information on streams they have explicit access to. Unauthorized stream IDs will not be disclosed in order to prevent attacks.
- Stopping a Stream: This endpoint allows an authorized administrator to terminate an ongoing stream. Users cannot stop streams, ensuring that only privileged entities control the audio streams.

Web Application

The web application is the beating heart of the project. It serves as the central nervous system of the entire platform, facilitating real-time audio stream processing and distribution through the sophistication of multicasting. This web application is built off of modern technologies being NodeJS and ExpressJS. Through these modern tools, we are able to implement comprehensive security measures including HTTPS encryption, input validation, rate limiting, and detailed audit logging. The system maintains optimal performance through error handling. These features work in concert to ensure reliable, secure operations while supporting multiple concurrent users.

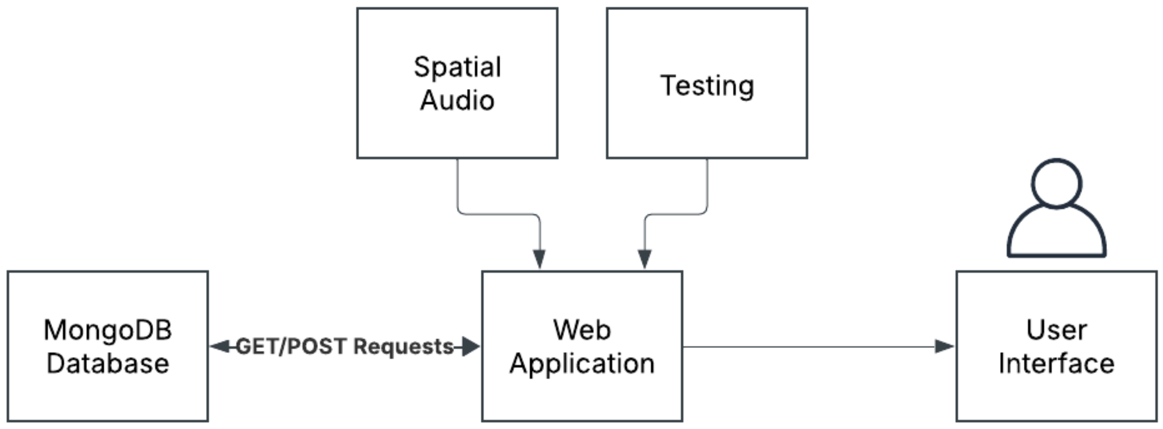


Figure 4: Web Application Component Diagram

Database

The database is responsible for storing user credentials and the system configurations made by the user. For the foundation of our data, MongoDB is the main data management strategy. It is chosen for its robust document-oriented storage capabilities and flexible schema design. The database architecture contains two primary domains: user management and system configuration. The user management handles credentials, roles, and authentication tokens. The system configuration stores audio channel definitions and interface customization.

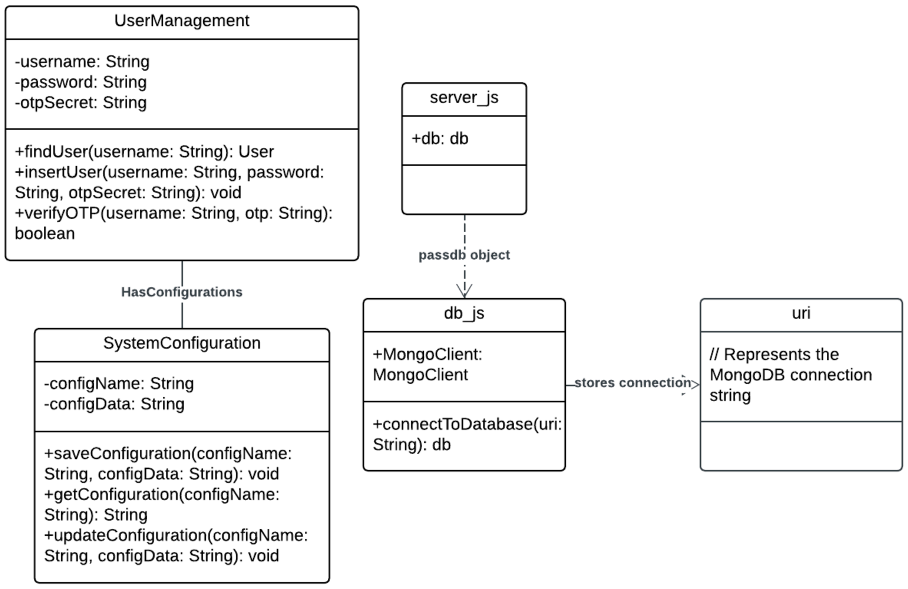

UserManagement
-username: String
-password: String
-otpSecret: String
+findUser(username: String): User
+insertUser(username: String, password: String, otpSecret: String): void
+verifyOTP(username: String, otp: String): boolean
HasConfigurations
SystemConfiguration
-configName: String
-configData: String
+saveConfiguration(configName: String, configData: String): void
+getConfiguration(configName: String): String
+updateConfiguration(configName: String, configData: String): void
server_js
+db: db
passdb object
db_js
+MongoClient: MongoClient
+connectToDatabase(uri: String): db
stores connection
uri
// Represents the MongoDB connection string

Figure 5: Database UML diagram

User Interface

The user interface emphasizes operational efficiency through a thoughtfully designed layout that reduces cognitive load while maintaining functionality. The design incorporates clear navigation and responsive elements that adapt to various devices seamlessly. The key features that make our project stand out are the ability to quickly select which channel you would like to listen to, seamless use of 3D audio to determine the location of your channel, comprehensive user controls for profile and preference management, and detailed monitoring tools for audio levels and system status.

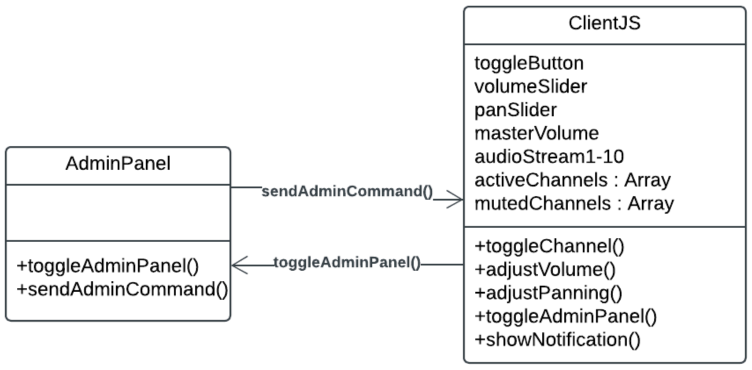


Figure 6: User Interface UML diagram

Spatial Audio

The spatial audio system provides a new revolutionary approach to General Dynamics current solution. This solution eliminates the need for a single physical speaker per each channel quickly eliminating the cost of hardware. If a user has 40 channels then they would have to buy 40 speakers. This architecture supports 360-degree sound positioning with distance-based attenuation and room acoustics modeling. Performance optimization ensures minimal latency and efficient resource utilization while maintaining audio quality across various output devices.

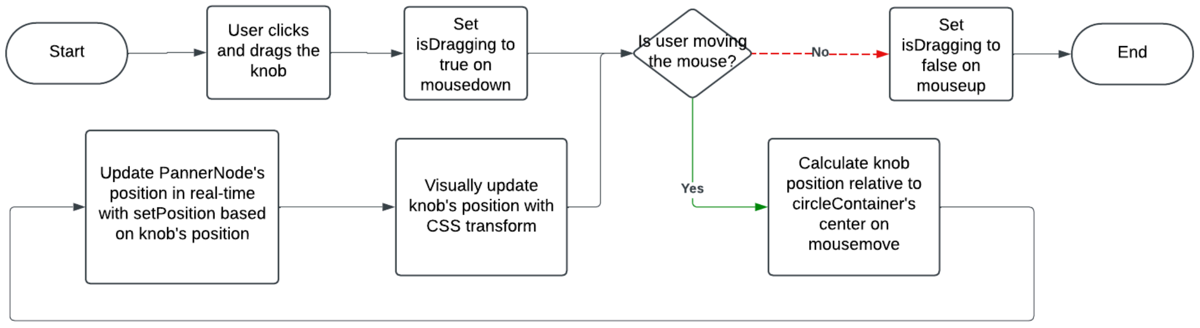

Start
User clicks and drags the knob
Set isDragging to true on mousedown
Is user moving the mouse?
No
Set isDragging to false on mouseup
End
Yes
Calculate knob position relative to circleContainer's center on mousemove
Update PannerNode's position in real-time with setPosition based on knob's position
Visually update knob's position with CSS transform

Figure 7: Spatial Audio Workflow

Testing

Testing is responsible for ensuring the functionality of the web application. Testing ensures that the application is staying on the right track and that the code fails properly when exceptions occur that are beyond the scope of the function. Testing is the quality assurance of the project. Quality assurance is maintained through a comprehensive testing framework that operates across multiple levels. Unit testing validates the individual components and functions. For the unit testing, JEST was the one that was chosen as it provides code coverage and standard testing and GitHub integration making testing significantly easier for everyone to see and understand. System-level testing ensures the end-to-end workflow validation and performance under stress conditions. The testing infrastructure automates execution within a continuous integration pipeline. Quality assurance processes include mandatory code reviews, coverage metrics, and security scanning to maintain high standards of reliability which is all built into the GitHub commit process.

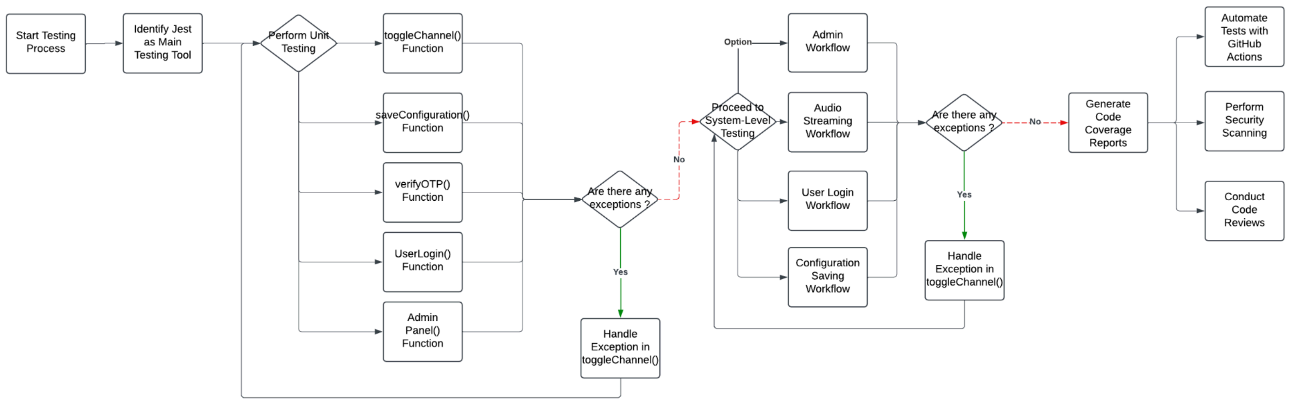

Start Testing Process
Identify Jest as Main Testing Tool
Perform Unit Testing
toggleChannel() Function
saveConfiguration() Function
verifyOTP() Function
UserLogin() Function
Admin Panel() Function
Are there any exceptions ?
Yes
Handle Exception in toggleChannel()
No
Proceed to System-Level Testing
Option
Admin Workflow
Audio Streaming Workflow
User Login Workflow
Configuration Saving Workflow
Are there any exceptions ?
Yes
Handle Exception in toggleChannel()
No
Generate Code Coverage Reports
Automate Tests with GitHub Actions
Perform Security Scanning
Conduct Code Reviews

Figure 8: Testing workflow

Security

The zero trust security framework operates on the principle of “never trust, always verify” implementing continuous authentication and verification at every level of system interaction. Identity verification incorporates constant password verification and MFA authentication. Data protection measures include end-to-end encryption and comprehensive access audit logging. Application security implements runtime monitoring, while compliance measures ensure regulatory alignment and policy enforcement. The framework maintains security through continuous monitoring and automated response ensuring legitimate user access and denying unauthorized users access.

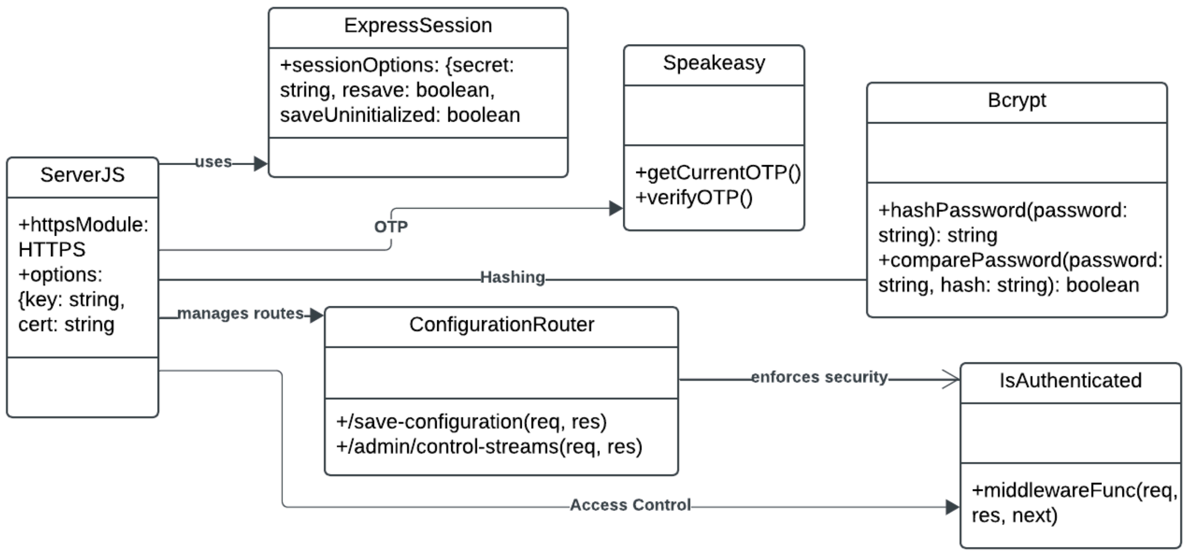
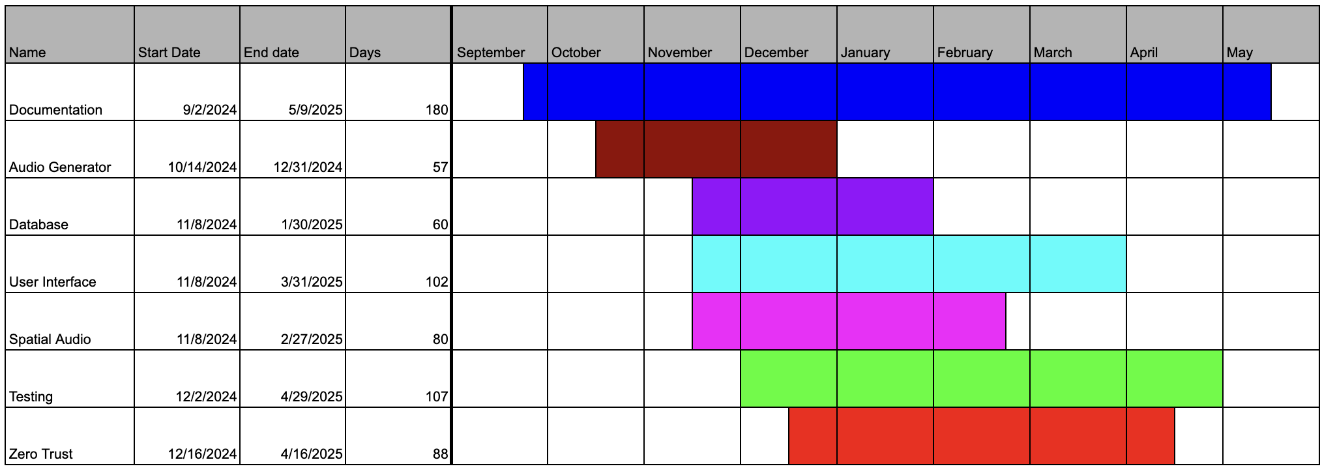

ExpressSession
+sessionOptions: {secret: string, resave: boolean, saveUninitialized: boolean
Speakeasy
+getCurrentOTP()
+verifyOTP()
Bcrypt
+hashPassword(password: string): string
+comparePassword(password: string, hash: string): boolean
ServerJS
+httpsModule: HTTPS
+options: {key: string, cert: string
uses
OTP
Hashing
manages routes
ConfigurationRouter
+/save-configuration(req, res)
+/admin/control-streams(req, res)
enforces security
IsAuthenticated
+middlewareFunc(req, res, next)
Access Control

Figure 9: Security Workflow

Implementation Plan

The implementation of this project follows a structured timeline consisting of various programming practices and phases. Our general approach to development is a combination of individual sprints, pair programming and weekly internal meetings. With individual sprints we can make rapid progress towards our assigned modules, leaving time for feature testing and validation. By pair programming, we can work through tougher programming problems and fix bugs. Lastly, with internal meetings the entire StratoSplit team is up to date on what's going on within development, what needs to be done moving forward, and who might need help with their respective projects.



Name	Start Date	End date	Days	September	October	November	December	January	February	March	April	May
Documentation	9/2/2024	5/9/2025	180	■	■	■	■	■	■	■	■	■
Audio Generator	10/14/2024	12/31/2024	57		■	■	■					
Database	11/8/2024	1/30/2025	60			■	■	■				
User Interface	11/8/2024	3/31/2025	102			■	■	■	■	■		
Spatial Audio	11/8/2024	2/27/2025	80			■	■	■	■			
Testing	12/2/2024	4/29/2025	107				■	■	■	■	■	
Zero Trust	12/16/2024	4/16/2025	88				■	■	■	■	■	

Figure 10: Project Implementation Timeline: Development Phases and Scheduling Overview

- **Documentation:** Beginning with planning in September 2024 and extending until May 2025, documentation plays a critical role throughout the development cycle, ensuring that all technical aspects are well-documented and continuously updated. This phase lays the foundation for the project's architecture and system design. All team members are responsible for this module.
- **Audio Generator:** Development for the audio generator module began in October 2024. It is responsible for generating audio streams from MP3 files to be sent over multicast groups using RTP. The initial phase of this component was completed in December 2024 and has been integrated into the system, with potential future updates. Nolan is responsible for this module.
- **Database:** Work on the database began in November 2024. The database is essential for storing system configurations and user data. This has been completed and integrated as of December 2024. Dallon is responsible for this module.

- **User Interface:** The first version of the user interface was developed in November 2024. This component provides the interactive front-end for users and will continue to be developed until March 2025 to allow for iterative improvements. All team members are responsible for this module.
- **Spatial Audio:** Initial spatial audio implementation begins in November 2024 with directional panning. Development will continue until February 2025 and will be improved to implement ambisonic sound processing, ensuring that 3D audio rendering is optimized before testing and integration. Sam is responsible for this module.
- **Testing:** Testing began in December 2024 and will continue through April 2025. This phase includes unit tests, integration tests, code coverage calculations, and performance evaluations to ensure a stable and reliable system. Elliot is responsible for this module.
- **Zero Trust:** In mid December, Zero Trust security measures began to be implemented and are scheduled for completion by April 2025, reinforcing security policies across the system. All team members are responsible for this module.
- **Deployment:** The final phase from April to May 2025, focuses on finalization and Deployment. This period is dedicated to final testing, bug fixes, and documentation updates, ensuring a seamless transition to the deployment phase. The completion of documentation in early May 2025 marks the official conclusion of the development cycle, signifying the project's readiness for launch.

This structured timeline ensures that critical components are developed in a logical sequence, allowing dependencies to align effectively. The phased approach with overlapping development, testing, and security implementation ensures that potential issues are identified early, leading to a robust and scalable system.

Conclusion

Developed in collaboration with General Dynamics Mission Systems, StratoSplit directly addresses the inefficiencies of existing radio modem interfaces by providing a scalable, secure, and user-friendly web application. With a focus on simulated audio generation, real-time transmission, and an intuitive monitoring dashboard, our solution ensures that operators can efficiently manage and interact with radio communications in dynamic environments.

At its core, StratoSplit was designed to address the fundamental challenge of inefficient outdated communication interfaces used in modern radio systems. Existing solutions often make it difficult for operators to effectively manage and transmit audio data in mission-critical environments. These limitations can lead to delays, miscommunication, and operational inefficiencies, particularly in high-stakes scenarios such as search-and-rescue missions and military operations. By developing a secure, scalable, and intuitive web application, we overcome these challenges by streamlining audio generation, optimizing real-time transmission, and providing an interactive dashboard for monitoring and control. This ensures that users can access, manage, and interact with communications seamlessly, reducing response times and improving overall operational effectiveness.

The system’s architecture—built on cloud-based infrastructure, containerized applications, and a robust MongoDB backend—emphasizes high availability, security, and scalability. By implementing zero trust security principles and leveraging AWS GovCloud, we have designed StratoSplit to withstand evolving cybersecurity threats while maintaining optimal performance under varying network conditions. Features such as spatial audio, modular development, and rigorous testing ensure that the final product is both innovative and reliable.

Throughout the structured development timeline, we have focused on iterative testing, security enhancements, and continuous refinement of the user experience. By addressing potential risks such as tool incompatibilities, service downtimes, and integration challenges, we have established mitigation strategies to ensure system stability and security.

Ultimately, StratoSplit delivers a transformative communication solution that modernizes audio transmission for mission-critical operations. By aligning with the requirements of GDMS and the U.S. Coast Guard, our team is committed to delivering a high-performance platform that enhances operational efficiency and ensures seamless communication for its users.