Spatialisation As A Service (SaaS)

Final Project Report

Callum John Spiller

Programme of study:
BSc FT Computer Science (Apprenticeship)

Supervisor:
Johan Pauwels

Final Year
Undergraduate Project 2022/23

Queen Mary University of London

School of Electronic Engineering and Computer Science
April 16, 2023

Abstract

Forming part of a degree apprenticeship classification, this paper investigates the efficacy of public cloud infrastructure in meeting business goals and service-level agreements through the design and development of web applications. A proof-of-concept spatial audio processing pipeline is developed and deployed in a public cloud environment. This paper documents and assesses these activities.

The rationale behind this paper derives from the sponsoring company, Sony Interactive Entertainment (SIE), and their business goals. The application developed as a part of the project attempts to address the problem of accessibility to spatial audio technology; and improve the ability of SIE's customers to preview technology offered by their PlayStation 5 (PS5) console.

A literature review is conducted and covers topics of spatial audio, cloud infrastructure, and web technology. A project timeline is planned, executed, and an analysis of the project's successes and failures takes place.

The project finds success in moving compute-heavy audio processing tasks into the public cloud and circumventing restrictive hardware requirements for experiencing spatial audio. This paper also details the various challenges and issues with developing in the public cloud, in addition to the strengths and shortfalls of the project's execution.

Contents

Glossary

Agile A framework or a collection of methods through which software projects can be conducted. Defined by its focus on incremental and iterative development practices. 24

AWS Amplify An AWS product that hosts web applications. Integrating with code repositories, it supports the use of Continuous Integration/Continuous Deployment (CI/CD) 25

AWS API Gateway An AWS service that allows developers to create and deploy their own custom APIs that allow access to data and application logic. 24, 25

AWS Elastic Container Registry An AWS function that allows the user to host and deploy container images. 25, 37, 38

AWS Eventbridge A cloud service that allows the communication between disparate AWS serverless components through the use of events. 24

AWS Lambda Part of Amazon's serverless computing services, these Lambda functions allow a software developer to host and run individual fragments of code that are triggered in response to events. The feature manages the resources required for these pieces of code to run. 24, 25, 28

AWS Simple Queue Service A message queue service from AWS that allows the sending and receiving of messages without data loss. 25

AWS Step Function A service that allows a software developer to orchestrate and schedule serverless workflow pipelines through the use of state machines and tasks. Through this service, a developer is able to easily control the use and order of other AWS services. 24, 25

container image A standalone package of sotware that contains everything needed in order to run an application. This allows for the identical deployment and running of applications independant of the underlying infrastructure. 25

DevNet A private web resource accessible only to licensed PlayStation developer, publisher and middleware companies. On this site you are able to access PlayStation API documents and order PlayStation development kits. 6

Docker A set of products that allows developers to virtually deploy operating systems and software packages in reuseable containers. Docker is one of the most widely used prodcuts for this purpose. 25

game development kit Game development kits are use to develop software that is intended to run on a specific game console. They provide the hardware and software needed in order to do this. 6

Node.js A free and open-source server environment that allows the execution of JavaScript code outside of a web browser. In this project it was used to build the front-end web interface. 24

PlayStation Partner The name given to any video game developer or retailer who engages with the PlayStation ecosystem in order to sell or develop their products. 6, 8, 21

React.js A free and open-source Javascript library, maintained by Meta, that is used primarily for developing front-end user interfaces by rendering components to the DOM in web browsers. 24–27

spatial audio Is the manipulation of sound produced by stereo speakers to mimic the human localization of sound sources in three-dimensional space. 6, 7, 24

Tempest 3D Audio Engine The proprietary audio engine used by the PlayStation 5 in order to render ambisonic audio files to headphones. 6, 22

three.js A JavaScript library that can be used to build and render 3D content in a web browser. 27

Acronyms

API application programming interface 6, 7, 11

AWS Amazon Web Services 7, 13–15, 17, 21, 23–25

CI Continuous Integration 14

CI/CD Continuous Integration/Continuous Deployment 4, 18, 25

DSP digital signal processing 6, 7

EC2 Amazon Elastic Compute Cloud 25

GDPR General Data Protection Regulation 16

HRTF head-related transfer function 7, 10, 11, 22

IDE Integrated Development Environment 18

IP Intellectual Property 23

MUI Material User Interface 26, 28

MVP minimum viable product 15, 21, 26

PaaS Platform as a Service 9

POC proof of concept 7

PS5 PlayStation 5 6, 11, 23

S3 Amazon Simple Storage Solution 25

SDK Software Development Kit 24

SIE Sony Interactive Entertainment 6–8, 15, 16, 18, 20, 21, 23

SLA service-level agreement 6

UI User Interface 24

1 Introduction

1.1 Project Background

This project is a part of a degree apprenticeship hosted by SIE. This project must therefore consider the business domain of the sponsoring company.

With the release of the PS5 in November 2020, spatial audio technology has become a key product for SIE as game developers seek to leverage the Tempest 3D Audio Engine espoused by the video game console. In addition, SIE is shifting to cloud-based infrastructure[1] across the wider business in order to leverage the cost, flexibility, and reliability benefits that cloud technology affords [Qian et al., 2009].

In response to these changes within the company, this project aims to engage both of these emergent technologies in order to address one of SIE's major service-level agreements (SLAs) with a software solution.

1.2 Problem Statement

At the time of writing, the onboarding of Partners to the developer and publisher platforms managed by SIE is an area in the company that has been targeted for improvement. Getting Partners engaged with PlayStation's development and sales ecosystem faster and with less 'friction' has been outlined as a major SLA.

This report argues that, despite the fact that the Partner onboarding process has been improved, the ability for Partners to trial PlayStation's spatial audio technology is too restricted. There is currently a dependence on local hardware setup in order to experience PlayStation's spatial audio in an interactive manner.

Currently, Partners who want to experiment with Tempest 3D Audio Engine must perform the following steps:

1. Apply for and order a PS5 game development kit
2. Await delivery of the kit
3. Set up the game development kit
4. Research the application programming interfaces (APIs) documentation provided by SIE's DevNet

This process is sub-optimal for a Partner who wishes to get a quick insight into what is possible with spatial audio.

This project proposes an alternative solution where spatial audio can be accessible as a web service, effectively eliminating the need for a game development kit. The system would be accessible through a browser and will mean that the Partner will be granted easier access, fulfilling the aforementioned SLA.

1.3 Project Aims

The aim of this project is to research, design, and engineer a web service that allows a user to experience spatial audio in a way that reacts to their input. The intended complexity of this project concerns the cloud infrastructure design and implementation that forms the backbone of the *"Spatialisation-as-a-Service"* concept; the project seeks to move compute-heavy digital

[1]like many other companies relying on web technology [Qian et al., 2009]

signal processing (DSP) tasks from local hardware to a cloud environment. The project can therefore be classified primarily as an infrastructure project, and discusses the topics and challenges relating to this area.

The proposed workflow allows a user with a standard web browser[2] to upload their own music file[3] to a website and ultimately receive back a new audio file that demonstrates spatial audio as a customised transformation of the original file. This new audio file will be constructed from the isolated 'stems'[4] of the original file which will be separated out from each other as a part of the DSP pipeline. The user will then choose where to 'position' these stems in a virtual 3d space by setting spatial parameters for each stem. Finally, the rendering of the stems and their spatial parameters will take place using head-related transfer functions (HRTFs) to render a 'spatialised' version of the music file. All the audio processing will execute in Amazon Web Services (AWS) cloud environments to circumvent any hardware requirements and challenges.

For information security reasons, the prototype produced as a part of this project will *not* feature any proprietary SIE software and use only libraries and code that exist in the public domain. Because of this, the prototype can be considered a proof of concept (POC) where, if successful, SIE software might be transplanted into the serverless DSP pipeline.

1.4 Project Objectives

In order to achieve the aims set out in 1.3 the project must produce a number of deliverables:

1. A project plan which outlines the timeline of both the research and development of the project.
2. A review of pertinent literature relating primarily to:
   - Public Cloud infrastructure and services (especially those available from AWS)
   - Audio spatialisation
   - Web audio APIs and front-end technology
3. A review of existing stereo-to-spatial services and technologies.
4. A functioning stereo-to-spatial serverless pipeline.
5. A frontend that enables a user to interact with the conversion pipeline by uploading and downloading audio files, as well as setting parameters for conversion through a web API.
6. A testing framework that supports iterative development.
7. A report on user testing.
8. A review and analysis of how the produced system has met or missed the targets.

[2] Chromium-based browsers and Firefox
[3] Of either of the .mp3 or .wav format
[4] For example, we might separate a song into vocals, bass, and drums

1.5 Research Questions

In order to guide the research and development process of the proposed system, this report will seek to answer the following research questions:

1. What are the characteristics of audio-processing pipelines that are executed within cloud infrastructure?
2. What is the impact of cloud technology in addressing physical hardware limitations?
3. How effective is cloud infrastructure in facilitating the execution of compute-heavy audio pipelines?
4. How can the leveraging of cloud technology improve the experience of the users of SIEs Partner platform?

2 Literature Review

As outlined in 1.3, this project attempts to engage with cloud infrastructure, spatial audio processing, and web development frameworks. To approach these topics thoroughly, this report performs a review of pertinent literature. In doing so, this report considers and incorporates existing ideas in these fields while providing a foundation from which to answer the research questions listed in 1.5.

2.1 Cloud Computing

Ever since internet service providers began the commercialization of cloud computing, it has become one of the major trends in the technology space [Qian et al., 2009]. According to Qian et al. [2009], 'cloud computing' is one of the most vague terms when it comes to the description of the technology on account of the breadth of its application

s2[Dillon et al., 2010]

Cloud computing technology is dominated by three major players, each with their own style of cloud services:

1. Amazon's Web Services which began as a means of server virtualization [Amazon, 2022]

2. Google's Cloud Platform, described by Qian et al. [2009] as a technique-specific sandbox that calls itself a Platform as a Service (PaaS) [Alphabet Inc, 2022]

3. Microsoft's Azure Network [Microsoft, 2022]

2.2 Audio Spatialisation

2.2.1 Origins

Blauert [1996] notes in their seminal text, *Spatial Hearing: The Psychophysics of Human Sound Localization*, that: "human beings are primarily visually-orientated", and that the other senses are less developed in comparison. This difference has been mirrored in the history of scientific research. Wade and Ono [2005] note that research in binaural hearing was developed later than binocular vision partially due to the difficulty in controlling the audio stimuli in experiments. It was only later on that the concept of distinction between the *sound event* and the *auditory event* as influenced by binaural hearing became prevalent. Blauert [1996] explains that this distinction informs the practice of audio replication analogous to its originating sound event:

> The telecommunications engineer, of course, is especially interested in just those cases in which the positions of the sound source, and the auditory event do not coincide. The telecommunications engineer seeks to reproduce the auditory events that occur at the point where a recording or transmission originates, using the smallest possible number of sound sources (e.g., loudspeakers) [Blauert, 1996].

The patent filed in 1958 by Alan Dower Blumlein[5] details an early stereophonic system, which exploits the human sound localization ability for the enhancement of entertainment

[5] [Blumlein, 1958]

experiences[6] [7]. Blumlein observed that in film theatres there was a certain level of cognitive dissonance whereby the actor's voices sounded like they were coming from a different location than where they appeared on the screen [Alexander, 1999]. This patent, in response, specifically outlines methods for introducing stereophonic audio to sound film as a means of increasing the perceived "quality" of the entertainment experience. Blumlein acknowledges that human binaural hearing is responsible for the ability to localize sound, and his patent is an example of how one might induce an auditory event that exhibits spatialisation on the horizontal plane through the control of inter-aural time differences [Blumlein, 1958].

The patent marked an improvement in the way that auditory events might be replicated by introducing this form of spatialisation, and the vestiges of Blumlein's ideas can be observed in modern spatial audio techniques [Politis et al., 2017, Beyer and Raichel, 1999]. What is perhaps the most salient aspect of the document, however, is that it recognizes the physiological factors that are involved in human sound localization. These physiological factors explored and expounded upon by Blauert [1996], and, as noted in the 1996 revision of his book, become more important as audio spatialisation and entertainment technology attempts to induce auditory events that imply three-dimensional audio spaces.

2.2.2 From two to three

> The external ears superimpose linear distortions on the incoming signals, which, in each case, are specific for the direction of incidence of the sound wave and the source distance. In this way, spatial information is encoded into the signals that are received by the eardrums [Blauert, 1996].

Roginska and Geluso [2017] note that: "the word 'binaural' refers, at the most basic level, to hearing with two ears, but it later came to include all the spatial cues from the ears, head, and body of a listener". Binaural recordings can therefore refer to the practice of capturing sounds that incorporate human physiology. This is executed with dummy mannikin heads with microphones placed inside the ears so that sound entering them are affected by the 'blocking' nature of the head; developments in this technology rapidly sped up throughout the 20th century [Paul, 2009]. While other forms of spatial representation were developed in this period [Gerzon, 1973, Noisternig et al., 2012, Berkhout et al., 1993], technology that considers the physical and physiological factors in human listening when attempting to induce auditory events that feature sound localization.

Roginska and Geluso [2017] identify that while capturing binaural audio is relatively easy, realizing the same effect through post-recording production is considerably harder and poses the challenge of modelling the human spatialization facility.

2.2.3 Getting the head in the game

The head-related transfer function (HRTF) can be described as a representation of the perceptual cues that facilitate human sound localization as a sound propagates from its source to the human ear [Suzuki et al., 2011]. This modelling of the human sound localization facility allows for this HRTF to be applied to a sound before reaching the human eardrum [Roginska

[6] This author acknowledges that this is not the first example of this kind of system; the control of inter-aural time differences was pioneered by Clément Ader as early at 4 years after Bell's invention of the telephone. This was for the purpose of rendering a spatial transmission of the Paris Opera. This further solidifies a history of the desire for spatial immersion in entertainment.

[7] There is a rich history of considering space in the composition of music in Western Classical tradition, with Italian renaissance composers writing for *cori spezzati*, or multiple choirs that are spatially separated [Morucci, 2013]. This author mourns that the topic of spatialisation in historic acoustic performance goes beyond the scope of this report.

and Geluso, 2017]. It is with this technology that more and more modern entertainment systems begin to localize sounds [Blauert, 1996, Honda et al., 2007, Roginska and Geluso, 2017, Suzuki et al., 2011, Xie et al., 2013, Cerny, 2020, Hong et al., 2017] during audio playback.

There are many software systems, toolkits, and frameworks that have been developed to allow engineers to build software that can utilise HRTFs and apply them to monophonic recordings [Cuevas-Rodríguez et al., 2019, Gorzel et al., 2019]. It is through these technologies that many video game systems such as the PS5 are able to provide immersive 3D audio experiences. In commercial systems such as these, consideration must also be applied to the selection of the HRTF that are used. While each person's experience of sound is as individual as they are, capturing the HRTF of each individual who engages with the product is not yet feasible due to the highly involved and costly process of capturing them. Considerable research has been done in order to develop and produce HRTF databases that appeal to a wide variety of subjects, taking into account individual and non-individual HRTFs [Armstrong et al., 2018]. It is common practice to have entertainment systems contain multiple HRTF options to choose from when setting up that system's spatial audio capabilities [Shukla et al., 2018].

2.3 Web Audio

Audio-visual media on the internet is extremely widespread and its delivery takes a myriad of forms [Brügger and Milligan, 2018]. The means by which audio is delivered to users on the internet is most frequently through a web audio API, the most common of which is the one developed by Mozilla [W3c, 2021, Mozilla, 2019]. One of the major benefits of utilising web technology in combination with audio technology is that it allows developers and those who wish to present audio to an audience to do so with a rich toolset of graphical libraries that are easily accessed through a web browser [Pauwels, 2018]; this is most frequently seen in commercial usage through web audio players such as Spotify and SoundCloud as a natural evolution of the radio broadcasting format [Bottomley, 2020]. Web frameworks such as React.js[8], Flask[9], and Django[10] are all capable of handling and displaying audio from a web page.

Audio delivery is primarily executed through the downloading and playing of a static file or as a packet stream from a server-based audio file source; however, more recently web audio can be delivered peer-to-peer in real-time through such technologies like WebRTC [Ünver et al., 2020, García et al., 2019].

[8] [Minnick, 2022]
[9] [Zhai et al., 2022]
[10] [Pauwels, 2018]

3 Risk Assessment

A risk is defined by the Project Management Institute [2021] as "an uncertain event or condition that, if it occurs, can have a positive or negative effect on one or more objectives" [11]. The effective management of project risk across a number of risk-management frameworks involves the prior identification of said risks [Goman, 2021]. For this project, a risk assessment is performed across three categories as a means of improving the likelihood of the project meeting its objectives as laid out in 1.4.

This assessment takes the form of three tabular risk registers with columns evaluating each risk's impact and likelihood, as well as preventative actions taken as a result of this identification. These risks will be subject to ongoing monitoring, and are subject to change over the duration of the project's completion.

3.1 Project Risks

These risks affect the project's schedule and affect the project's ability to be finished within a given timeframe.

Table 1: Project Risks

Risk	**Impact**	**Likelihood rating**	**Impact rating**	**Preventative actions**
Failure to access required information	Lack full understanding of the background material	Low	Medium	Be diligent in identifying alternative sources, as well as making use of the Queen Mary Library's resource-purchasing facility
Scope creep	During development the scope of the project may increase and what is attempted goes beyond what is realistically capable over the duration of the project	Medium	Low	Clearly define the scope at the outset of the project, have a roadmap in place and be accountable for sticking to it
Low productivity	When work on projects slow, any delays can cascade and cause the project to miss its objectives by the end of the project duration	Medium	Medium	Be diligent in creating and sticking to a project plan - additionally communicate frequently with the project supervisor in order to be accountable and to resolve any issues promptly

[11] [Project Management Institute, 2021]

Lose access to AWS	Given that the project is hosted in the cloud, losing access to administrate the service would result in a severe delay in development	Low	High	Make sure that all credentials are up to date before starting development - check company policy surrounding credential expiry
Loss of work	If the codebase is lost during development then this means having to re-write all of the code, slowing down the project dramatically	Low	Medium to High	Ensure that codebase management systems are used - this means using a version-control system such as GitHub to store all code and documentation for the project - also ensure that any AWS deployments have backups
Inefficient working	When time is spent on menial, small, or administrative tasks that do not directly contribute to the project's completion, this can cause progress on the overall project to slow	High	Medium	Ensure that tasks are prioritised effectively, deploying agile scrum effort ratings when needed
Lacking requisite skills	When a challenge in the project requires skills that this author does not possess then they must spend time learning the skills required to overcome the challenge - this can slow progress on the project	Medium	Medium	When planning the project timeline, ensure that enough leeway has been granted to tasks to allow for extra time spent on learning and development
Unexpected levels of complexity	Only when projects are begun do certain challenges arise - the project may become far more complex and disorganised than originally planned	Medium	Low	As well as planning effectively, this author can ensure that they are diligent when it comes to researching the technologies they use - this means that they are able to develop the project effectively

3.2 Product Risks

These risks pose threats to the quality, or performance of the prototype.

Table 2: Product Risks

Risk	Impact	Likelihood rating	Impact rating	Preventative actions
Insufficient prototype testing	The product does not meet functional and non-functional requirements	Medium	High	Specify a framework for testing as soon as possible in the development process, automate testing where possible using a CI pipeline
AWS instability	In the event of the cloud hosting and processing services going down, the product's service will be unavailable	Low	High	Make use of different availability zones within AWS so that in the event of failure in a single area, the project can be spun up again elsewhere
Code issues	If the project contains code that lacks quality, then bugs and unstable performance may cause the product to fail	Medium	High	Conform to best-practice coding standards, frequently test code in regression and unit tests, ensure any bugs are promptly patched
Insufficient research	When a product is made without properly researching the best methods to do so, that product can be insufficient in comparison to competition in a business environment	Low	Medium	Ensure that enough time is scheduled to experimenting with different technology and researching pertinent literature before proceeding with development
Poor design	If the product has been badly designed then it will not meet the desired level of quality and user requirements - the product may even fail to work at all	Medium	Medium	Ensure that sufficient time has been given to the planning stage - in the event that unforeseen problems come to light ensure that help is sought to resolve the design issue and rebuild if necessary.

Poor project management	If the project is managed poorly then the product may not be built to a satisfactory level - delays and lack of proper oversight can cause a drop in the quality of the delivered product	Low	Medium	Ensure that the project management software is correctly set up - use alerts for deadlines and make an effort to work on the project often
Unrealistic project goals	If the targeted MVP is not of appropriate scope then the product might not get finished in time for the project deadline in the event that there is too much work to do	Low	High	Work in an agile and iterative fashion - start small on tasks and gradually build up complexity
Insufficient resources	If the project does not get the compute power or cloud resources it needs then the product's performance will be lacking or it may not function at all	Low	Medium	Have a clear outline as to what resources are needed and allow enough time to get the requisite permission from the company's AWS administrators

3.3 Business Risks

This project is undertaken as a part of a degree apprenticeship and associated business risks are present:

Table 3: Business Risks

Risk	**Impact**	**Likelihood rating**	**Impact rating**	**Preventative actions**
Unauthorised use of proprietary materials	Business-critical materials are leaked and cost SIE competitive advantage	Low	Critical	Never use any material developed by SIE as a part of business activity, use only open-source libraries
Large AWS fees	The project causes cloud fees charged to SIE to spiral out of control, costing the company far more than budgeted for	Medium	Medium	Make proper use of the AWS cost centre, define a budget, and set limits and alerts for budget usage in the AWS console

Cyber-security attack	In the event that the cloud app has a security vulnerability, the rest of the SIE tech stack may be at risk of being compromised	High	High	Utilise domain allow-listing, parameterization of user credentials, and routinely check the vulnerabilities of external dependencies - additionally, develop a breach response plan
Data mis-handling	In the event that the product stores user data that it does not have permission to, then the company may be liable for severe penalties under the GDPR[12]	Low	High	Keep stored user data to a minimum - in the case that user data is required, ensure that it is handled and disposed correctly, obtaining permission to do so

[12] IT Governance Privacy Team [2019]

4 Project Plan

4.1 Plan Preparation and Communication

To develop an effective plan for the project, the key deliverables for the project[13] were analysed and roughly estimated based upon their complexity. Each deliverable was given a broad estimate of the time this would take in proportion to that complexity, with more complex tasks, and tasks that were prone to delay, given more time allocated to them. These estimates were then mapped to the timescale of the project; project deadlines were already in place and deliverables were given target dates that were set in relation to these deadlines. Additionally, the more complex deliverables were broken down into subtasks and milestones to provide a more granular and manageable path to their completion. Text-based deliverables were given dates by which certain chapters needed to be completed, and development tasks were given dates by which that part of the application should be built and deployed by.

This project makes use of the ClickUp[14] software to perform project management functions and to organize workflows. This software has been chosen over other pieces of software in the education[15] and project management[16] space on the basis of cost and ease-of-use, as well as its ability to synchronize across different platforms. In addition, ClickUp was also used to communicate the project's progress with the project supervisor who was able to see the project plan and its progress over time through the updating of subtask statuses.

Prioritizing communication with the supervisor was integral to the success of the project because of the accountability and oversight it provided. Having a platform such as ClickUp drastically reduced the likelihood of error in communication and time management. A meeting with the project supervisor was scheduled every two weeks in order to identify and remedy issues with the project's execution.

4.2 Timeline

The below series of figures (1, 2, 3) detail the project's tasks and subtasks, and the planned timeframes for their completion. Timeframes were adjusted in accordance with the perceived complexity of each task following enlightenment from the literature review and market research. Dependencies for each task were also calculated and can be seen represented as an arrow in the timeline.

4.3 Resources

This project intended to use minimal resources in its development. As outlined in 2, one of the many advantages of cloud computing is its flexibility and ease of resource management. Given that the application would be hosted entirely in cloud environments, there would be no direct hardware costs associated with the project. The costs that do apply will relate to the use of the AWS platform. These costs needed careful management, as explained in 3.3.

Other resources utilized included:

1. Queen Mary library resources
2. The project supervisor

[13] As outlined in 1.4
[14] [ATD Staff, 2022]
[15] [Badaru and Adu, 2022]
[16] [Phipps, 2022]

3. Knowledge sharing from colleagues at SIE
4. JetBrains' Integrated Development Environment (IDE) Suite
5. Open-source audio-processing libraries.
6. Online articles and tutorials

4.4 Methodology

Murray [2016] notes that the "traditional" manner of software development follows a linear path from requirements, to design, to execution, to testing, and so on. This method often results in inflexibility when it comes to software project execution. This is because of its dependence on the full set of requirements being gathered before the development process begins. Any issues with the requirements (incompleteness, inaccuracy) using this method are typically only found at the end of the process, instead of along the way through a cycle of development and feedback[17]. This project is of relatively small scope, and the number of shareholders are small on account of its proof-of-concept status. As such, it serves to incorporate some, but not all, aspects of the 'Agile'[18] development methodology into the process:

> An Agile project starts with only the most high-level requirements. Sometimes these are referred to as "user stories." Such a requirement might sound like, "A user will be able to buy a subscription to our product on a new e-commerce website." There are no designs, no specifications.[19]

While this project will not go so far as to remove the need for a design or specification altogether, the foundation for the project's execution will rest on high-level user stories and identified functional requirements. In addition, the project will require frequent testing as development progresses. The intention, therefore, is to endeavour to implement a CI/CD pipeline to ensure any updates to the prototype can be tested and deployed in an online environment as it is being developed.

[17] [Murray, 2016]
[18] [Beck et al., 2001]
[19] [Murray, 2016]

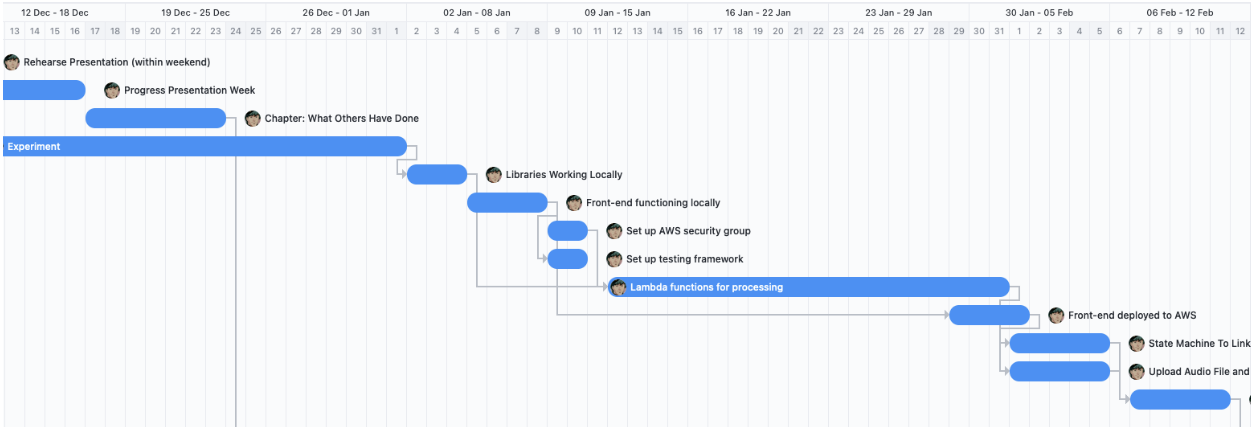

Figure 1: Timeline: $Dec\ 12th \rightarrow Feb\ 12th$

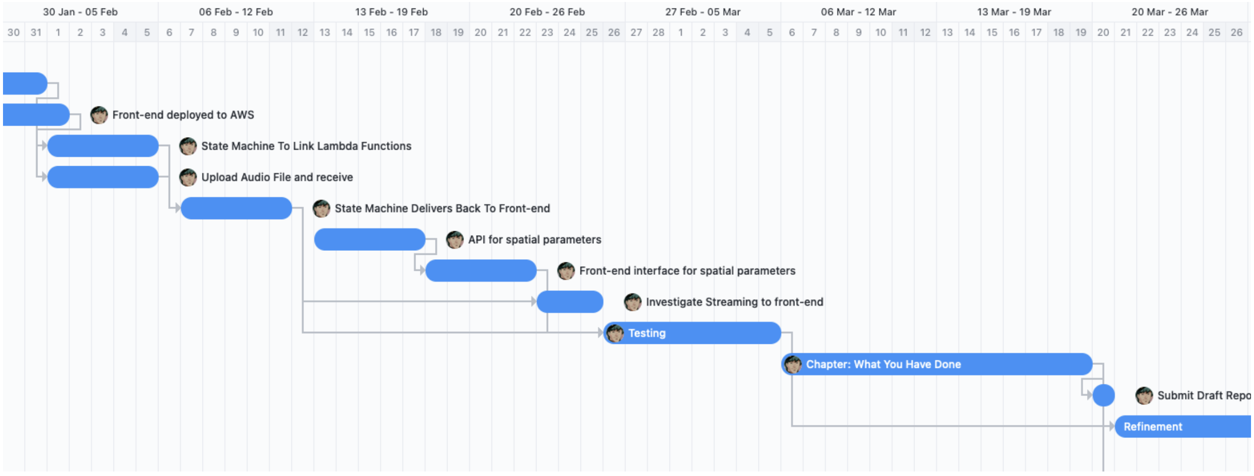

Figure 2: Timeline: $Jan\ 30th \rightarrow Mar\ 26th$

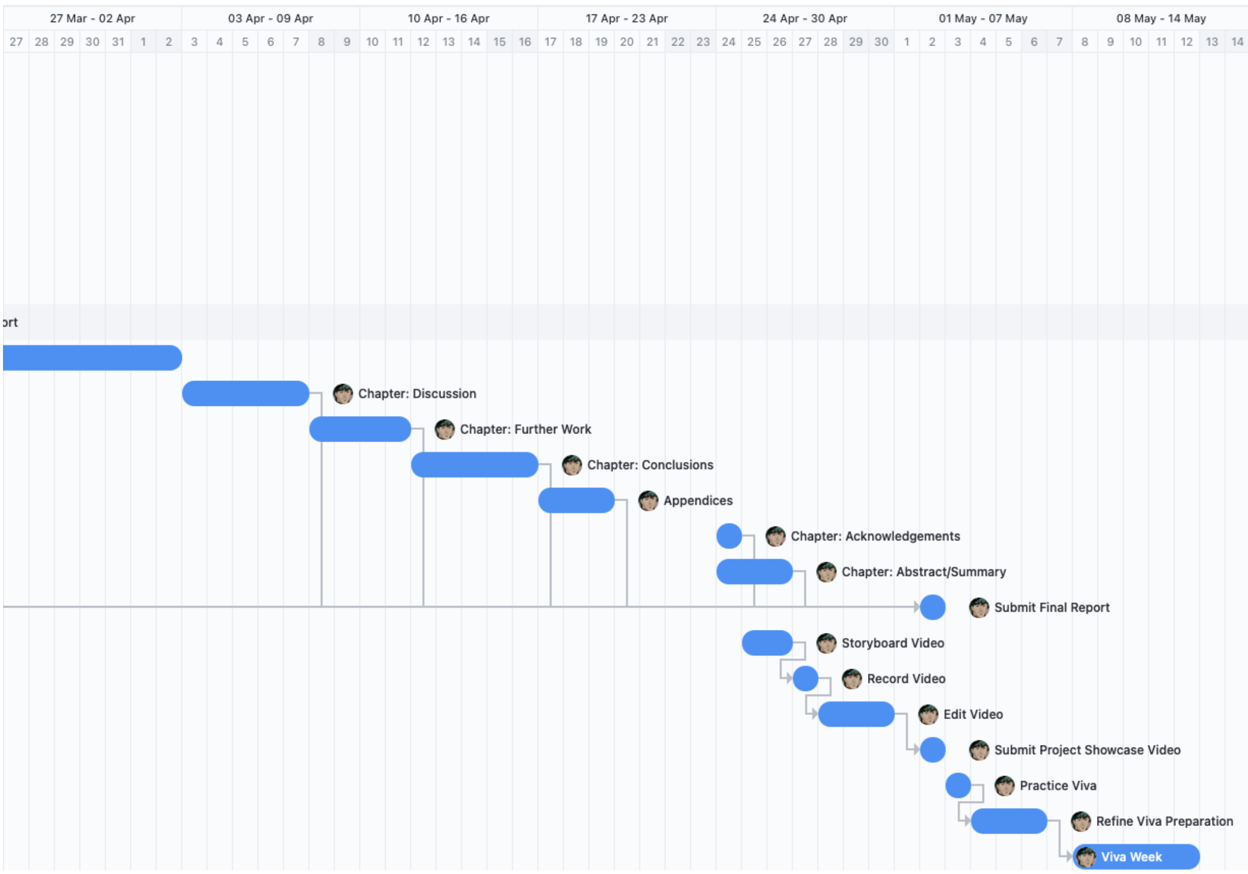

Figure 3: Timeline: $Mar\ 27th \rightarrow May\ 14th$

5 Requirement Capture and Analysis

According to Anton [2003]: “good requirements planning means software can be cheaper to produce, easier to build, and less prone to unexpected failure.” Requirements for this project considered the business, user, and functional requirements in different scenarios where the product might be used [Wiegers, 2000, Potts et al., 1994]. The requirements specification additionally considered the different types of people who would be using the product. It is the combination of the gathered requirements that informs the product’s design.

5.1 The Case For Business

Wiegers and Hokanson [2023] state in their new book that enterprise software product requirements stem from the core business requirements. These business requirements inform resultant system, user, and solution requirements. See Figure 4 for Wiegers’ and Hokanson’s representation of this influence. In addition, due to the nature of enterprise, certain business rules and constraints will influence the solution requirements. SIE’s PlayStation is a high-profile brand so additional considerations must be taken when operating in this space. See Table 4 for the business requirements that help inform the other requirements of the project.

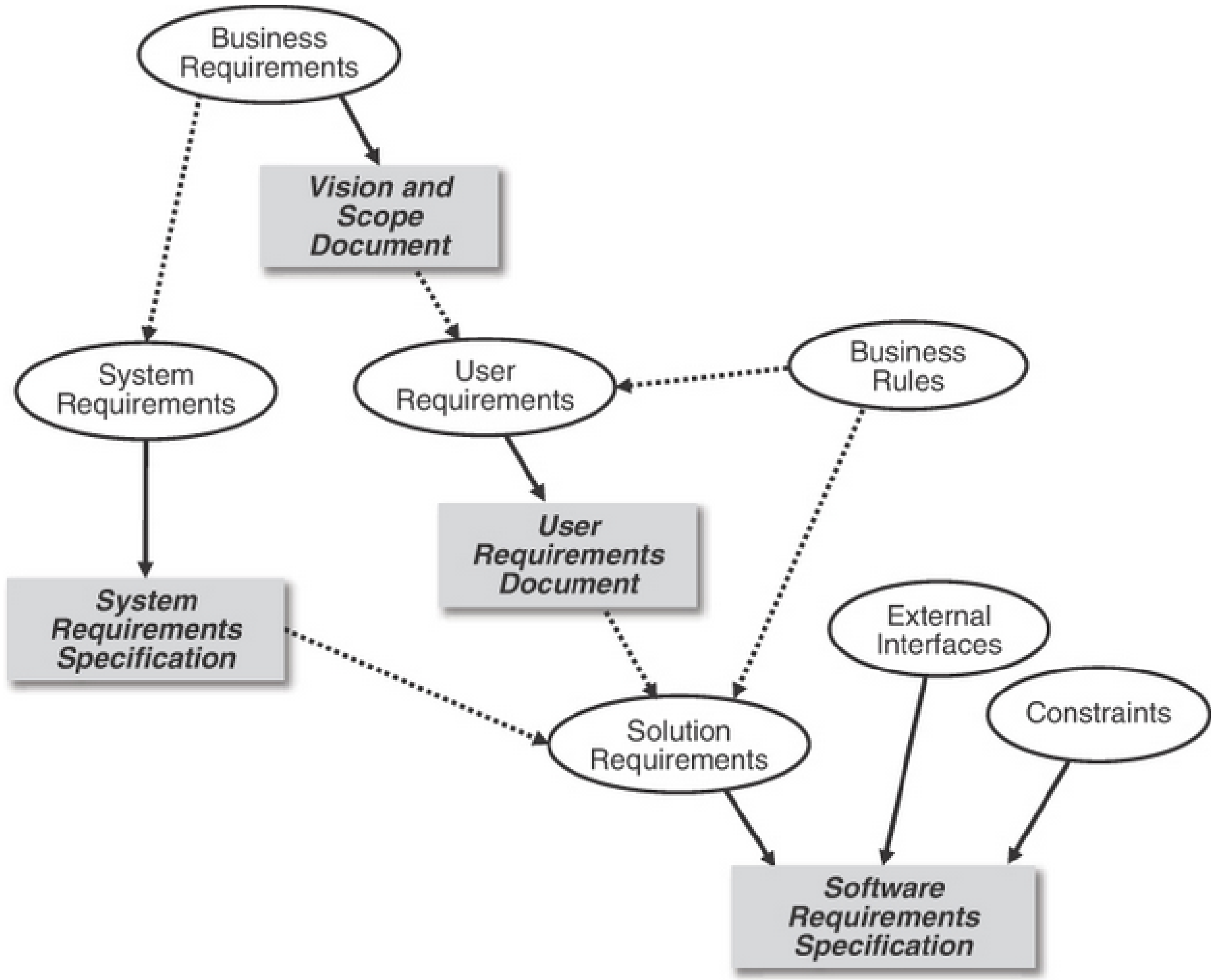

Figure 1.1 *Connections between several types of requirements information and containers that store them. Solid lines mean “are stored in.” Dotted lines mean “are the origin of” or “influence.”*

Figure 4: Taken from [Wiegers and Hokanson, 2023]

Table 4: Business Requirements

Code	Business Requirement
B1	Increase the accessibility to spatial audio technology for Partners
B2	Inform Partners and potential Partners of the nature and capabilities of spatial audio
B3	Increase the engagement of Partners with PlayStation's platform
B4	Demonstrate how spatial audio relates to the interests of Partners
B5	Increase the availability and flexibility of SIE's technical services

5.2 System Requirements

The problem statement in 1.2 makes things easy for the formulation of the key *system* requirements[20] for this system. This project defines the MVP in terms of the minimum level of functionality the system must have in order for the project to be considered successful. These requirements can be found in Table 5 and cover the technical requirements for the solution in relation to the business requirement.

Table 5: System Requirements

Code	System Requirement	Business Requirement Code(s)
S1	The system shall be accessible to the user through the use of a standard internet browser	B1
S2	The system shall allow the user to upload their own audio file	B2, B3
S3	The system shall separate the uploaded audio file into its constituent instrument stems	B2
S4	The system shall provide the means for the user to define spatial parameters for each stem	B2, B3
S5	The system shall render a spatial audio file taking the instrument stems and the user-defined spatial parameters as input	B2
S6	The system shall deliver the rendered audio file to the user through the web browser	B1, B2
S7	The system shall exist entirely within the AWS public cloud ecosystem	B1, B5

[20] Wiegers and Hokanson [2023] define system requirements as: "A description of a top-level capability or characteristic of a complex system that has multiple subsystems, often including both hardware and software elements."

S8	The system shall use HRTFs to render spatial audio, mimicking the functionality of the Tempest 3D Audio Engine	B2, B4, B5

5.3 User Requirements

The functional needs of the system are simple to pin down, however, the user requirements require slightly more nuance. The concept of 'user stories' has faced criticism in the literature for over-focus towards the 'story', rather than the 'user' and their role [Hudson, 2013]. Hudson [2013] highlights 'persona stories' as an alternative that takes into account the nature of the user: we ask ourselves the question: "what does this person, in this role, require from the system?"

5.3.1 Dramatis Personae

Three personae were formulated as potential users of the project, each with their own background and motivations:

1. Game Developer—wants to develop a game using spatial audio—they not yet onboarded with PlayStation partners, and want to see a demonstration of what the PlayStation platform can offer in terms of its spatial audio capabilities—this will inform their future choice of development platform.
2. Audio Enthusiast—has prior experience with stereo audio—wants to experience spatial audio applied to their favorite song.
3. Novice—stumbled across the feature—no prior experience of spatial audio or even game development—likes immersive media experience so feels an interest towards the concept of spatial audio.

The user requirements were formulated with these three personae in mind. The product is intended to be a customer-facing one so consideration towards a wide range of potential customers were considered.

Table 6: User Requirements

Code	User Requirement	Persona(e) Addressed	Business Requirement Code(s)
U1	The user shall be able to understand and use the system easily	Novice	B1
U2	The system shall have high-quality audio rendering quality	Game Developer, Audio Enthusiast	B2, B4
U3	The user shall be informed by using the system	Novice, Game Developer	B1, B4

U4	The system shall be easier set up and run than a local environment	Game Developer	B1, B5
U5	The system pipeline shall complete quickly	Novice, Audio Enthusiast, Game Developer	B3, B5

5.4 Business Rules...ok?

Finally, the system's design must also take into account the operation of SIE as a company in its design:

- The proprietary spatialisation software that is used by the PS5 is a company secret, therefore the system should not use it in this prototype.
- SIE is concerned with being profitable, therefore costs should be minimised in the development and running of the service.
- The creation of the prototype should not be done in the SIE AWS environment.
- SIE retains tight control over the PlayStation Intellectual Property (IP) therefore its branding should not be present in the prototype.

These business rules will inform the design and the development practises of the project, particularly in relation to security and cost management.

6 Infrastructure & Design

Figure 5 details the initial design for the project's cloud architecture. The diagram provides a high-level overview of the primary user journey, and displays the AWS products that were intended to fulfil this journey. However, given that this project is executed using elements of the Agile methodology,this design was subject to change as development progressed. This section aims to describe how the initial design evolved to accommodate the changing system requirements of the project.

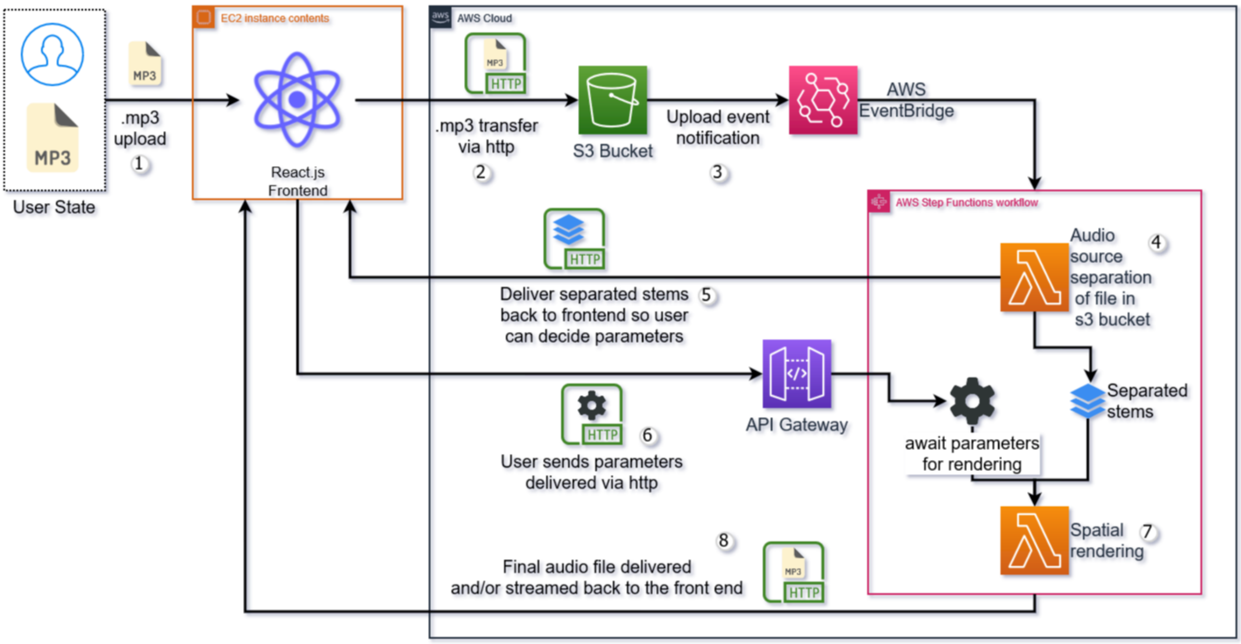

Figure 5: Preliminary Design for Cloud Architecture

6.1 Best Laid Plans

The product is designed for a wide range of users with varying knowledge of spatial audio. In order to accommodate as many potential users as possible, a web-hosted User Interface (UI) is considered essential. The user's primary mode of interaction with the service would be through a website built using the React.js front-end library. This library was chosen because of its ubiquity in modern web engineering; the depth of documentation available, and existing familiarity with the framework.

The front-end was designed to make use of the AWS Software Development Kit (SDK) for Node.js. This would allow the interface to initiate and interact with the audio processing pipeline hosted in AWS.

Figure 5 displays the two AWS Lambda functions that would comprise the backbone of the audio processing pipeline. Furthermore, these AWS Lambda functions would be coordinated and executed using the AWS Step Function service. This service was highly desired for this purpose since the processing pipeline would follow a series of discrete steps that require input from the user. AWS Step Functions would be able to accommodate this easily.

The entire service was initially intended to be coordinated solely through the use of the internal messaging service: AWS Eventbridge. This service, in conjunction with AWS API Gateway, was intended to form the most of the communication between the back and front ends of the application.

6.2 Refinement

As development progressed; it became apparent that there were issues with the initial design.

The first problems arose with the development of the AWS Lambda functions that would be used to process the audio. AWS Lambda functions require a deployment package to be created and then uploaded to AWS in order for the function to have the resources it needs to run. Unbeknownst to this author, there is a limit of 50 MB on the size of the deployment packages that could be uploaded directly to AWS Lambda. Given that the 3D Tune-In Toolkit and the Spleeter source separation libraries have significant dependencies that are more than 50 MB when zipped, a re-design needed to occur.

This issue was solved by making use of the AWS Elastic Container Registry service. This service allows a developer to host and deploy container images, crucially, with a size limit of 75 GB allowed for each image uploaded to the registry. Through the use of Docker container images, the code for the Lambda functions was bundled into deployment packages that could be uploaded to an AWS Elastic Container Registry and then linked to a Lambda function while staying well under the data limit offered by the container registry. The Dockerfiles for the source separation and spatialisation lambda functions can be found in Appendix A.1. Note the installation of the AWS Lambda runtime environment for both Python in Listing 1 and C++ in Listing 2.

The next major revision to the architecture design came with the removal of the proposed AWS API Gateway interface. Since the AWS Step Function requires input from the user to execute, the step function callback feature would be used to wait for an input message from the front end to the step function. The most elegant solution for this problem was to create a AWS Simple Queue Service queue through which messages could be sent and retrieved asynchronously without the risk of losing information through API communication failure. This was implemented through the creation of a unique message queue per interaction with the product. A unique identification key is created in the React.js interface, which is then used to identify both the AWS Simple Queue Service queue, and the resources uploaded and retrieved from the relevant Amazon Simple Storage Solution (S3) buckets. This key is not the same as a session key and would regenerate if the user refreshed the page to return to the start of the application.

With these changes, it became possible to define a final cloud architecture and workflow.

6.3 Finalising the structure

Figure 6 depicts the final infrastructure design of the project. The figure has been annotated with numbers to show the order in which data flows around the architecture. In addition to the changes mentioned above, there are a few other notable changes that are evidenced in the figure.

- The AWS Step Function has been fleshed out with the discrete steps of the workflow orchestration. An Amazon States Language representation of the function can be found in Listing 3.
- There are three S3 buckets to hold the artefacts from each stage of the process.
- The React.js frontend is now hosted in AWS Amplify instead of Amazon Elastic Compute Cloud (EC2); allowing for the implementation of CI/CD each time a commit is pushed to a branch in GitHub.

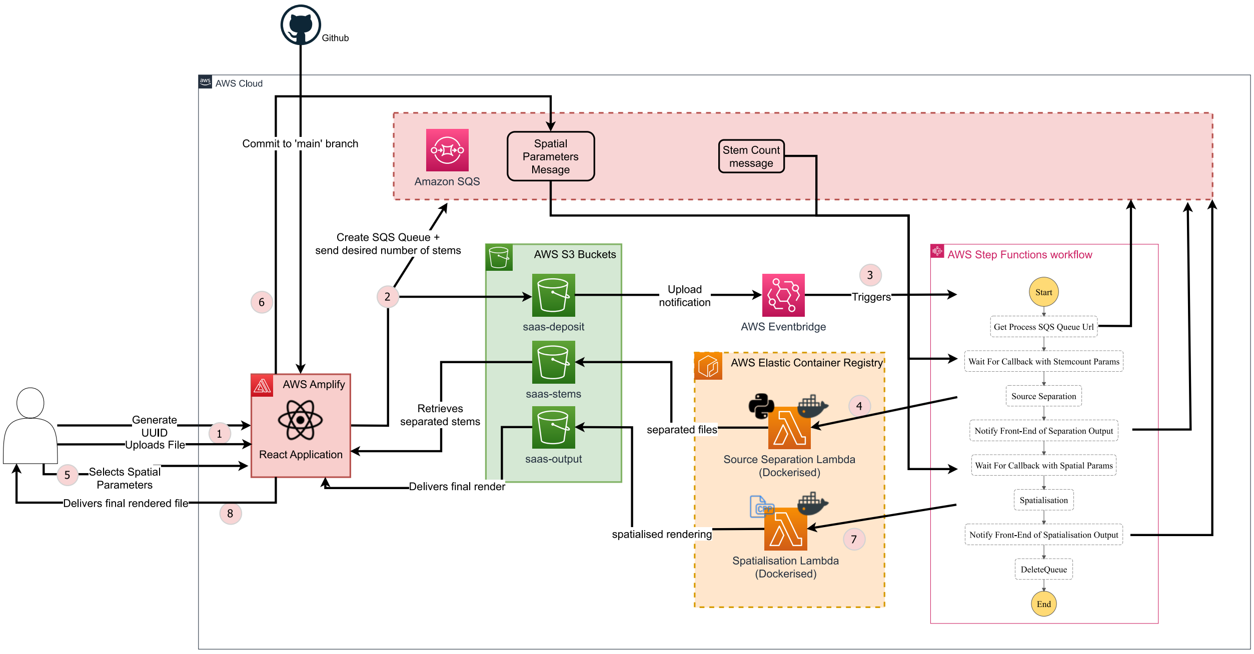

Figure 6: Final cloud architecture diagram

6.4 A fresh coat of paint

Early user testing of the product revealed that elements of the front-end interface needed improving[21]. The initial interface was basic with minimal styling applied to the React.js interface. See Figure 7 for a screenshot of the interface's early form.

Figure 7: Early implementation of application frontend

After the MVP was achieved, there was some additional capacity available to improve the frontend.

- The project was adapted to use tools provided by Material User Interface (MUI), a React.js component library[22], in order to better structure the project and improve the aesthetic feel. See Figure 8 for the improved welcome screen for the application.
- A slideshow was developed using the component library that explained the theory behind spatial audio in an easy-to-understand manner and explains what the product aims to do (Figure 9).
- The music file upload stage of the application was redesigned using MUI and turned into a three-step process using the 'Stepper' component (Figure 10).

[21]Further information on the testing and feedback process can be found in Section 8
[22]https://mui.com/core/

- The 'Spinner' component is used to help decorate the process of waiting for the stems to be separated out (Figure 11).

Figure 8: Refined splash screen that greets the user

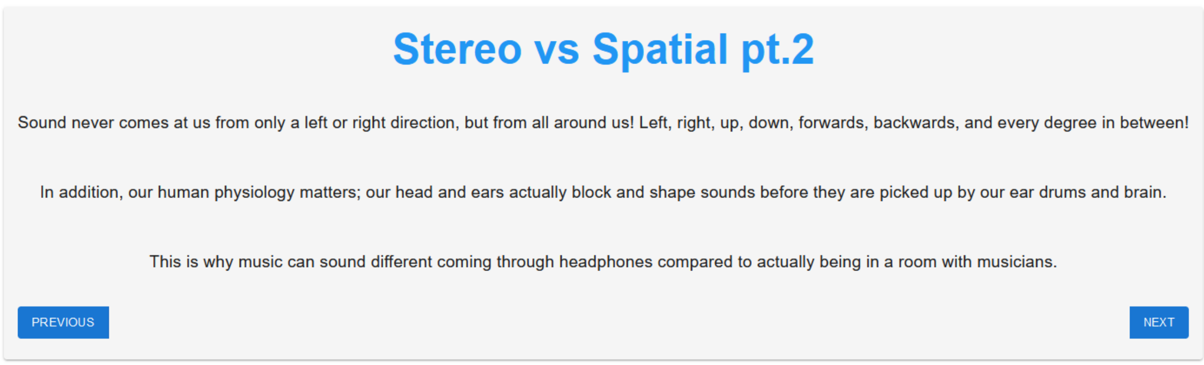

Figure 9: One of the slides from the introductory explanation of the project

6.5 No spatialisation without representation

An important piece of feedback on early demos of the product was that there was not enough visual or audio feedback on the user's input of the spatial parameters. As a result of this, the most significant redesign of the product allowed the user to experience and preview the effects of their input instead of being forced to wait for the final product to render and be delivered to the frontend. Using React Three Fiber, a React.js renderer for three.js[23], a 3D model represents the positioning of the audio stems in the virtual spatial field that the application eventually renders. Previously only a list of sliders were shown for each stem, but the final version lets the user see and hear what changes are being made in real-time as the stems

[23] [Dirksen, 2023]

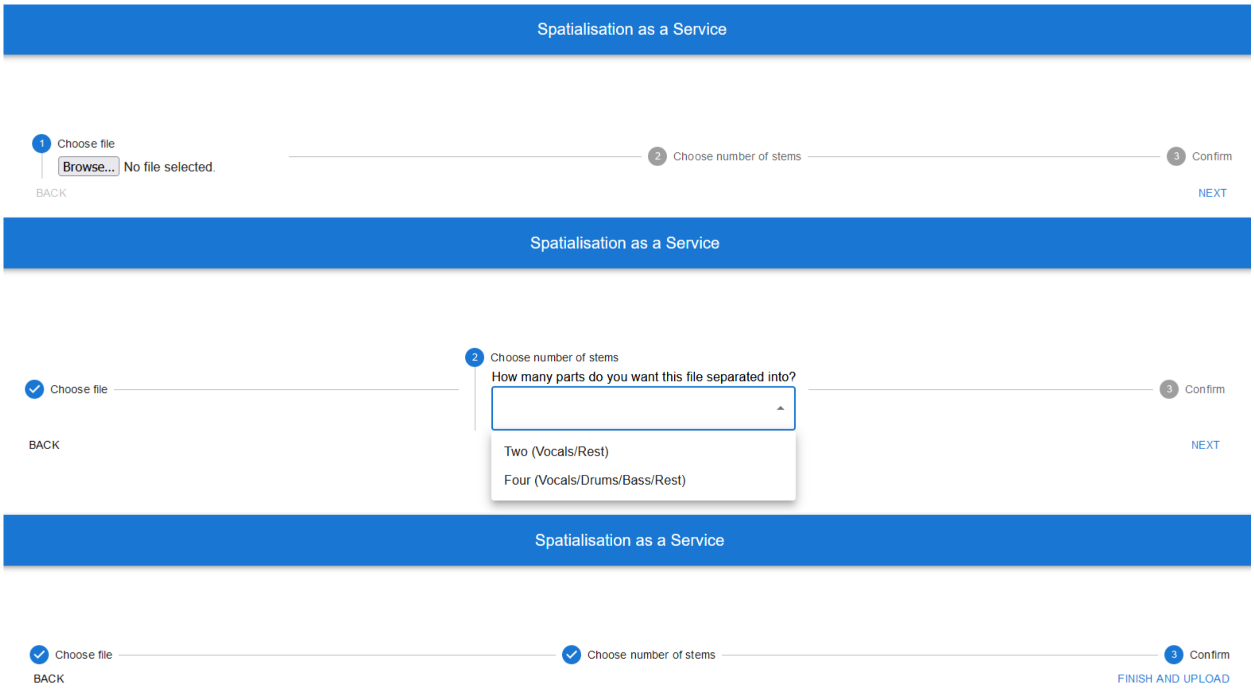

Figure 10: Audio upload stage remodelled using the MUI Stepper component

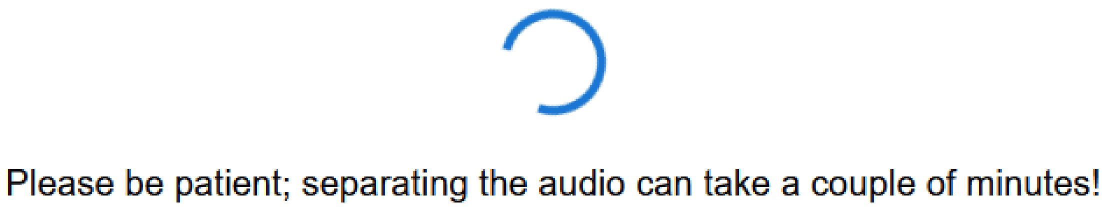

Figure 11: MUI Spinner and message displayed while audio stems are separated

are being played back. Figures 12, 13, and 14 demonstrate how the 3D model displays the user's input on screen. The user is able to rotate and zoom in and out of the model to see where the stems are positioned in relation to the listening position at the origin of the model. In addition, instead of the stems being played separately as before, the stems synchronise to each other and the user can mute or 'solo' each stem as they wish. This functionality is similar to that found in modern Digital Audio Workstations. The Howler.js library, which is used for audio playback, allows the accessing of the spatial audio module in the Web Audio API. Because of this, the user can preview the audible position of each stem while they are playing in the application. However, the quality of this preview is not as high-quality as the final rendered output that is created using the 3D-TuneIn API; this justifies the retention of the option to render the audio using the workflow hosted in AWS Lambda.

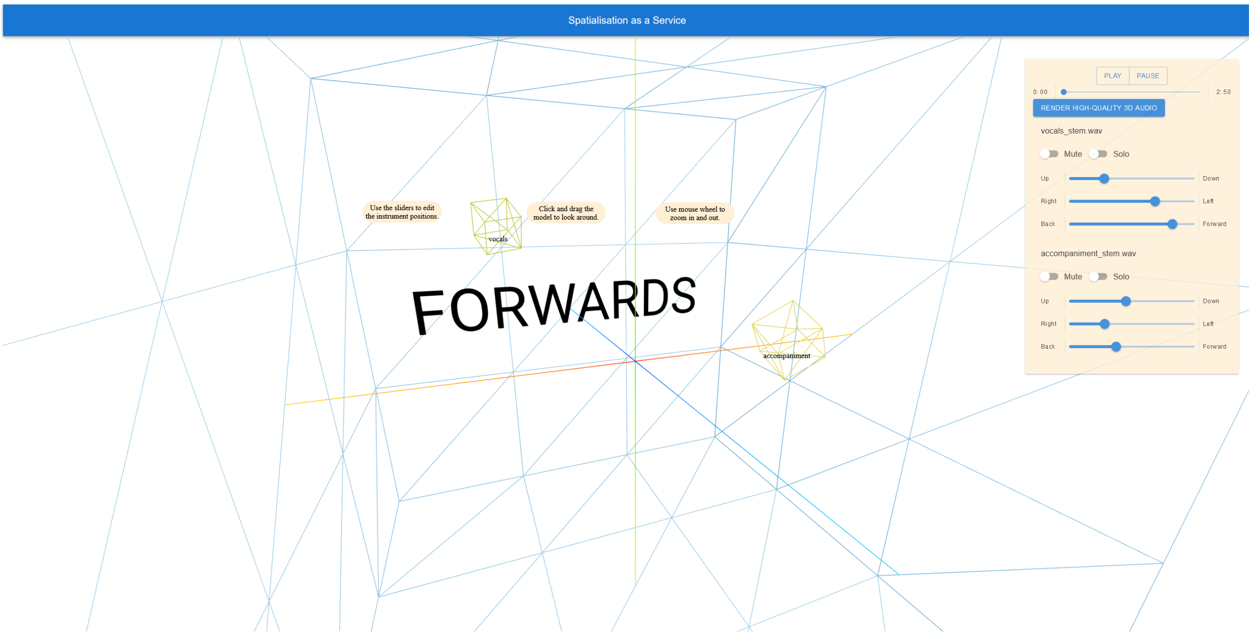

Figure 12: Perspective 1 of the 3D model interface

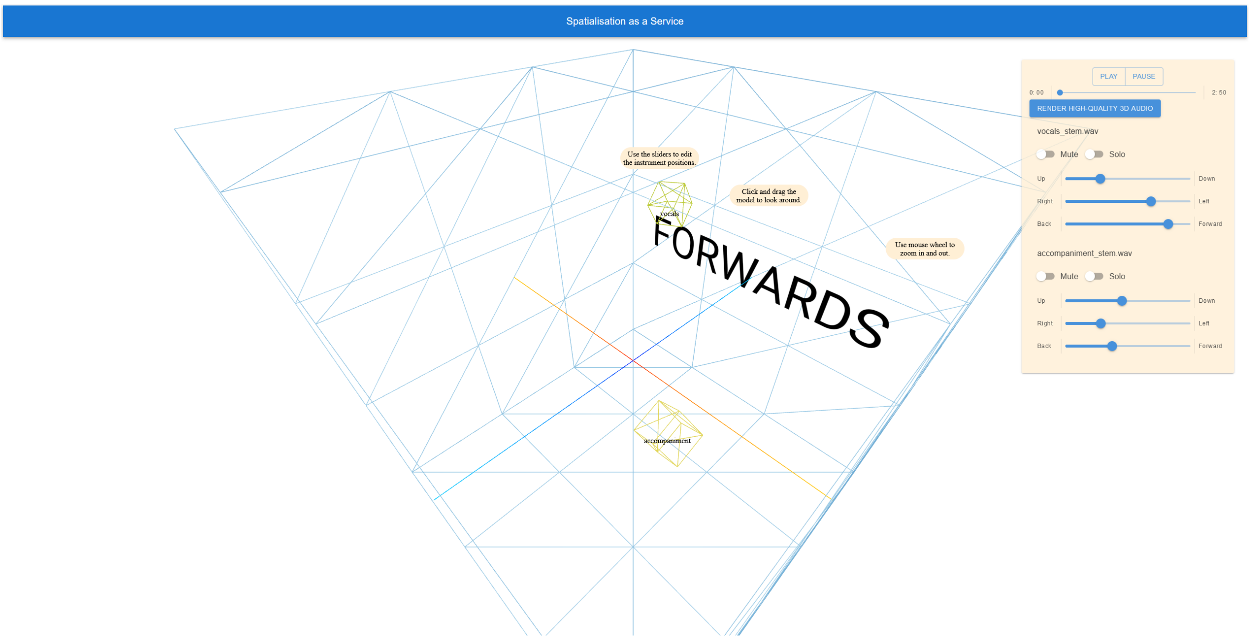

Figure 13: Perspective 2 of the 3D model interface

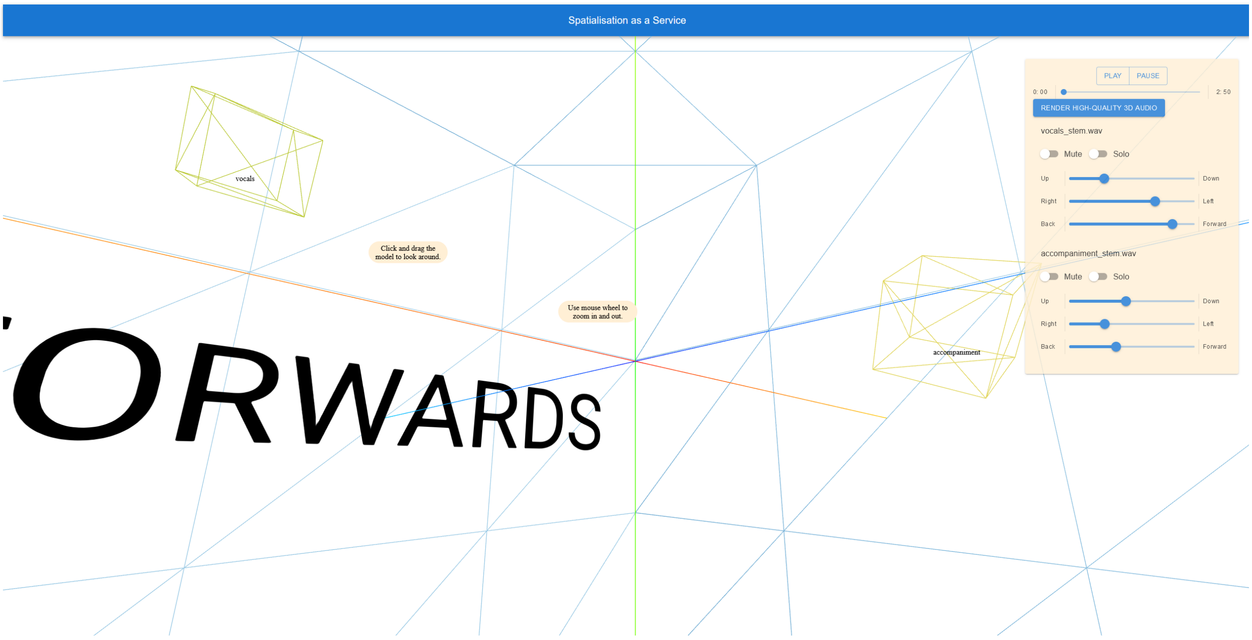

Figure 14: Perspective 3 of the 3D model interface

7 Implementation

– start small, project timeline aimed to reach MVP then refine - minimum requirements clearly defined, frontend not a priority – AWS threw up a few challenges - workflow consisted of working until a problem then consulting documentation for assistance – ci/Cd using aws amplify and GitHub integration. Combined with playwright unit tests for frontend and end-to-end tests in backend – made multiple branches and builds for testing purposes. Used a separate branch for developing and another to send out for user testing – recursive lambda function incident - called itself and resulted in a large bill which I needed to speak to customer service to get resolved – after results of user testing I needed to refine the product. Front end component was missing - much more time needed to be put into react.js, MUI and playwrights test writing to accommodate new react components at the unit test level – not so much feature creep than lack of time for proposed complexity - desired features required much more learning than previously thought, especially in docker builds and cmake structures – local builds proved difficult for Dev purposes, often had to wait for docker images to be built and uploaded and changes to react to be built and deployed using amplify, lots of dev time eaten up waiting

– code refactor and documentation towards end to maximise readability

8 Testing and Evaluation

Testing was a continuous process. Given the nature of the cloud environment things needed to be deployed in an environment so they could be tested.

This is why CI/CD was used. Commits and pushes to AWS triggered redeployments and the ability to test changes in a cloud environment quickly in addition to running regression tests each time.

Manual testing was required to ensure functionality was there but user testing was also used. A form and a link to an early build of the product were sent out and responses collected. These responses helped assess the effectiveness of the products functionality and guide future development processes.

Product was a success in not only delivering intended functionality but also extending it beyond the static spatial rendering but also preview the spatial audio in a real-time environment. Front end was far more in depth after refinement

9 Discussion

Success based on the initial criteria of the project. Met the core functional requirements and succeeded in creating a cloud-based service that is genuinely unique and allows easy access to trialling high-quality spatial audio to anyone with a browser and a music file.

Additionally, user feedback was very good - most were impressed with the core product despite having reservations about the user interface and its ability to convey information and instructions to the user.

However, the major issue with the project was in its inaccurate user requirements that needed redefining in the face of feedback. Initially was only concerned with the functionality for MVP. However didn't not fully consider the fact that it was real people using the project who may not initially understand what the software did. User experience was far more important than initially thought to user satisfaction. To the extent that some users did not understand the idea of spatial audio and confused the product with stereo music making software.

Given this, I did a good job of pivoting and, once the core functionality of cloud architecture was finished, spent a lot more time refining the user experience. This resulted in: a slideshow explaining the process, a much more refined ui, and even a live mockup of the results, with previews of the individual stems and able to change the parameters and hearths results in real time. While this feature did not have the same level of quality as the 3dti render, it vastly improved the user experience and showed how sources can be moved around in the virtual 3d space.

Overall I am very proud of the project and felt it hit all of the major requirements. I successfully managed to learn a bunch of new pieces of software and libraries, especially the services offered by AWS. I was able to avoid the normal pitfalls of waterfall development by adapting and meeting the need for altered requirements. The final software also showcases a good degree of skill in engineering, with code running efficiently and being well documented, in addition to following code style practises.

Given another few weeks of development I would have done better refactoring and separating the concerns of my front end code, however, given the complexity of the proposed product it is a great achievement to get a finished product. Also able to stay on time and project wasn't derailed even when falling behind (like trips to LA lol)

10 Further Work

Thankfully this project met a lot of the requirements, even the ones found after user testing.

However improvements could be made to the cold starts of the cloud rendering Initial separation of stems takes a long time because of cold starts and a way to reduce this would be most important could also add a load bar to track the progress of the lambda function would go a long way to I creasing user experience metrics

Demonstrate the capabilities of 3d toolkit better - allow changing of HRTF and BRIR profiles to the users tastes - this could be reflected in the 3d model also - it's good for basic demo but going a bit further would be great

Finally; improve UI further - make more detailed and precise.

The project had other stretch goals: have a user interface that allows the user to input values; expand the functionality by specifying HRTF files and SOFA environment files; allow for real-time previewing.

Stretch goal reqs:

Functional - Preview spatial rendering of individual stems - Real-time views of spatial audio – security - files are managed and secured to protect user info - fast response time in rendering - visual representation of the spatial audio rendering in 3d space - allow user to upload and choose SOFA and BRIR files - allow multiple types of audio file upload

Non functional – fully explains spatial audio and how it works - a smooth looking ui - take it beyond basic to a potentially customer-facing product - reliable - minimal bugs in the programming that result in failure for the end user. - edge case handling

11 Conclusions

Possible to perform complex and compute-heavy audio processing activities in a public cloud environment, even including user input to guide the process. Main drawbacks are the fact that processing time can take a while in some activities. For example, tensorflow being used for audio stem separation. However, the project proved that such an architecture is possible and improves accessibility to those without the requisit hardware.

As a promotional product success was mixed. Significant amount of UX work was required to produce a product that was customer friendly, and in the context of SIEE, this was not given as much consideration as required. Future developments would focus on these issues as well as improving the performance of the overall processing pipeline.

12 Acknowledgements

Hello there

A Appendix: Code Snippets

A.1 Lambda Function Dockerfiles

```
# Define global args
ARG FUNCTION_DIR="/function"
ARG BASE=python:3.9

# Pull base python image
FROM ${BASE} as build-image

# Install dependencies needed for lambda runtime API
RUN apt-get update && \
    apt-get install -y \
    g++ \
    make \
    cmake \
    unzip \
    libcurl4-openssl-dev

# Declare args for build

ARG SPLEETER_VERSION=2.3.2
ENV MODEL_PATH app/pretrained_models
ARG FUNCTION_DIR
RUN mkdir -p ${FUNCTION_DIR}

# Install pip dependencies from reqirements file
COPY requirements.txt .
RUN pip install \
    --target ${FUNCTION_DIR} \
    -r requirements.txt

# Copy app files
COPY app/ ${FUNCTION_DIR}

# Pull clean image for second stage
FROM ${BASE}

# Install dependencies needed for Spleeter
RUN apt-get update && \
    apt-get install -y \
    ffmpeg \
    libsndfile1
```

```

# Copy output from 1st build image into working directory
ARG FUNCTION_DIR
WORKDIR ${FUNCTION_DIR}
COPY --from=build-image ${FUNCTION_DIR} ${FUNCTION_DIR}

#Declare entry points for the function and the lambda handler command
ENTRYPOINT [ "/usr/local/bin/python", "-m", "awslambdaric" ]
CMD [ "app.handler" ]
```

Listing 1: Dockerfile for source separation Lambda function that is hosted in the AWS Elastic Container Registry

```
# Pull latest amazon linux image
FROM amazonlinux:latest
RUN yum update -y
RUN yum install -y yum-utils

# Install required tools
RUN yum -y groupinstall "Development tools"
RUN yum install -y gcc-c++ zlib-devel cmake3 python3-pip

# Specify compilers
ENV CC=gcc CXX=g++

# Install required tools
RUN yum clean -y metadata
RUN yum install -y libcurl-devel
RUN yum install -y openssl-devel
RUN yum install -y pulseaudio-libs-devel
RUN pip3 install awscli

# Fetch and install AWS Lambda Runtime for C++
RUN git clone https://github.com/awslabs/aws-lambda-cpp-runtime.git && \
    cd aws-lambda-cpp-runtime && mkdir build && cd build && \
    cmake3 .. -DCMAKE_BUILD_TYPE=Release \
    -DBUILD_SHARED_LIBS=OFF \
    -DCMAKE_INSTALL_PREFIX=/usr \
    && make -j8 && make install

# Fetch and install AWS C++ SDK (S3 and transfer libraries only)
RUN git clone https://github.com/aws/aws-sdk-cpp.git --recursive aws-sdk-cpp && \
    cd aws-sdk-cpp && mkdir build && cd build && \
    cmake3 .. -DBUILD_ONLY="s3;transfer" \
    -DBUILD_SHARED_LIBS=OFF \
    -DENABLE_UNITY_BUILD=ON \
    -DCMAKE_BUILD_TYPE=Release \
    -DCMAKE_INSTALL_PREFIX=/usr/local \
```

```
    && make -j8 && make install

# Fetch 3DTI toolkit and dependencies
RUN git clone https://github.com/3DTune-In/3dti_AudioToolkit.git 3dti_AudioToolkit && \
    cd 3dti_AudioToolkit/3dti_ResourceManager/third_party_libraries/ && \
    git clone https://github.com/USCiLab/cereal.git cereal && \
    rm -rf sofacoustics && \
    git clone https://github.com/sofacoustics/API_Cpp.git sofacoustics && \
    rm sofacoustics/libsofa/lib/libsofa.a

# Build 3DTI toolkit
RUN cd
↪   3dti_AudioToolkit/3dti_ResourceManager/third_party_libraries/sofacoustics/libsofa/build/linux
↪   && \
    make CONFIG=Release

# Fetch and install C++ logger
RUN git clone https://github.com/gabime/spdlog.git && \
    cd spdlog && mkdir build && cd build && \
    cmake3 .. && make -j && make install

# Copy program source files
RUN mkdir -p src
ADD src ./src/
ADD utils ./src/
COPY CMakeLists.txt ./src/
COPY config .aws/

# Build source files
RUN mkdir build && cd build && \
    cmake3 ../src -DCMAKE_BUILD_TYPE=Release \
    -DBUILD_SHARED_LIBS=OFF \
    -DCMAKE_INSTALL_PREFIX=/usr

#Build AWS deployment package
RUN cd build && make && make aws-lambda-package-spatialisation

# Unzip deployment package
RUN cd build && unzip spatialisation.zip -d unpacked

# Point to compiled function
ENTRYPOINT ["/build/unpacked/bin/spatialisation"]

# Uncomment line below for Docker Image Debugging
#CMD ["/usr/sbin/init"]
```

Listing 2: Dockerfile for spatialisation Lambda function that is hosted in the AWS Elastic Container Registry

```json
{
  "Comment": "State Machine for Spatialisation as a Service Application",
  "StartAt": "Get Process SQS Queue Url",
  "States": {
    "Get Process SQS Queue Url": {
      "Type": "Task",
      "Parameters": {
        "QueueName.$": "$.detail.object.key"
      },
      "Resource": "arn:aws:states:::aws-sdk:sqs:getQueueUrl",
      "ResultPath": "$.queueUrl",
      "Next": "Wait For Callback with Stemcount Params",
      "Comment": "Fetches SQS URL created by frontend on upload."
    },
    "Wait For Callback with Stemcount Params": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sqs:sendMessage.waitForTaskToken",
      "TimeoutSeconds": 1800,
      "Parameters": {
        "MessageBody": {
          "TaskToken.$": "$$.Task.Token"
        },
        "QueueUrl.$": "$.queueUrl.QueueUrl"
      },
      "Next": "Source Separation",
      "ResultSelector": {
        "val.$": "$"
      },
      "ResultPath": "$.stem-count"
    },
    "Source Separation": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "arn:aws:lambda:eu-west-2:024679833031:function:saas-source-separation:$LATEST",
        "Payload.$": "$"
      },
      "Retry": [
        {
          "ErrorEquals": [
            "Lambda.ServiceException",
            "Lambda.AWSLambdaException",
            "Lambda.SdkClientException",
            "Lambda.TooManyRequestsException"
          ],
          "IntervalSeconds": 2,
          "MaxAttempts": 6,
          "BackoffRate": 2
```

```
        },
        {
          "ErrorEquals": [
            "TypeError"
          ],
          "BackoffRate": 2,
          "IntervalSeconds": 1,
          "MaxAttempts": 2
        }
      ],
      "Next": "Notify Front-End of Separation Output",
      "ResultPath": "$.lambdaResult"
    },
    "Notify Front-End of Separation Output": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sqs:sendMessage",
      "Parameters": {
        "QueueUrl.$": "$.queueUrl.QueueUrl",
        "MessageBody": {
          "detail.$": "$.detail",
          "lambdaResult.$": "$.lambdaResult"
        }
      },
      "Next": "Wait For Callback with Spatial Params",
      "ResultPath": "$.notifyResult"
    },
    "Wait For Callback with Spatial Params": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sqs:sendMessage.waitForTaskToken",
      "TimeoutSeconds": 3600,
      "Parameters": {
        "MessageBody": {
          "TaskToken.$": "$$.Task.Token"
        },
        "QueueUrl.$": "$.queueUrl.QueueUrl"
      },
      "Next": "Spatialisation",
      "ResultSelector": {
        "spatialParams.$": "$"
      },
      "ResultPath": "$.params"
    },
    "Spatialisation": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "Payload.$": "$",
        "FunctionName":
        ↪  "arn:aws:lambda:eu-west-2:024679833031:function:saas-spatialisation:$LATEST"
```

```
      },
      "Retry": [
        {
          "ErrorEquals": [
            "Lambda.ServiceException",
            "Lambda.AWSLambdaException",
            "Lambda.SdkClientException",
            "Lambda.TooManyRequestsException"
          ],
          "IntervalSeconds": 2,
          "MaxAttempts": 6,
          "BackoffRate": 2
        }
      ],
      "Next": "Notify Front-End of Spatialisation Output",
      "ResultPath": "$.spatialisationOutput"
    },
    "Notify Front-End of Spatialisation Output": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sqs:sendMessage",
      "Parameters": {
        "QueueUrl.$": "$.queueUrl.QueueUrl",
        "MessageBody.$": "$"
      },
      "Next": "DeleteQueue",
      "ResultPath": "$.notifyResult"
    },
    "DeleteQueue": {
      "Type": "Task",
      "End": true,
      "Parameters": {
        "QueueUrl.$": "$.queueUrl.QueueUrl"
      },
      "Resource": "arn:aws:states:::aws-sdk:sqs:deleteQueue"
    }
  }
}
```

Listing 3: Amazon states language representation of the SaaS Step Function

References

Robert Alexander. *The Inventor of Stereo: The life and works of Alan Dower Blumlein.* Routledge, 1 edition, 1999. 10

Alphabet Inc. Google cloud overview, November 2022. URL https://cloud.google.com/docs/overview. 9

Amazon. Aws cloud essentials, 2022. URL https://aws.amazon.com/getting-started/cloud-essentials/?pg=gs. 9

Annie I Anton. Successful software projects need requirements planning. *Software, IEEE*, 20 (3):44–46, 2003. 20

Cal Armstrong, Lewis Thresh, Damian Murphy, and Gavin Kearney. A perceptual evaluation of individual and non-individual hrtfs: A case study of the sadie ii database. *Applied Sciences*, 8(11), 2018. ISSN 2076-3417. doi: 10.3390/app8112029. URL https://www.mdpi.com/2076-3417/8/11/2029. 11

ATD Staff. Clickup. *TD: Talent Development*, 76(6):9, 2022. ISSN 23740663. URL http://ezproxy.library.qmul.ac.uk/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edo&AN=157202341&site=eds-live. 17

Kazeem Ajasa Badaru and Emmanuel O. Adu. Platformisation of education: An analysis of south african universities' learning management systems. *Research in Social Sciences & Technology (RESSAT)*, 7(2):66–86, 2022. ISSN 24686891. URL http://ezproxy.library.qmul.ac.uk/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edb&AN=159642901&site=eds-live. 17

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development, 2001. URL http://www.agilemanifesto.org/. 18

A. Berkhout, Diemer Vries, and P VOGEL. Acoustic control by wave field synthesis. *J.Acoust.Soc.Am.*, 93:2764–2778, May 1993. doi: 10.1121/1.405852. 10

Robert T. Beyer and Daniel R. Raichel. Sounds of our times, two hundred years of acoustics. *The Journal of the Acoustical Society of America*, 106(1):15–16, 1999. ISSN 00014966. URL http://ezproxy.library.qmul.ac.uk/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edo&AN=ejs48802680&site=eds-live. 10

Jens Blauert. *Spatial Hearing: The Psychophysics of Human Sound Localization*. The MIT Press, October 1996. ISBN 9780262268684. doi: 10.7551/mitpress/6391.001.0001. URL https://doi.org/10.7551/mitpress/6391.001.0001. 9, 10, 11

A.D. Blumlein. British patent specification 394325. *Journal of the Audio Engineering Society*, 6(2):91, April 1958. 9, 10

A.J. Bottomley. *Sound Streams: A Cultural History of Radio-Internet Convergence*. University of Michigan Press, 2020. ISBN 9780472054497. URL https://books.google.co.uk/books?id=f9fkDwAAQBAJ. 11

N. Brügger and I. Milligan. *The SAGE Handbook of Web History*. SAGE Publications, 2018. ISBN 9781526455444. URL https://books.google.co.uk/books?id=kwJ6DwAAQBAJ. 11

Mark Cerny. The road to ps5, March 2020. URL https://www.youtube.com/watch?v=ph8LyNIT9sg&t=2304s. 11

María Cuevas-Rodríguez, Lorenzo Picinali, Daniel González-Toledo, Carlos Garre, Ernesto de la Rubia-Cuestas, Luis Molina-Tanco, and Arcadio Reyes-Lecuona. 3d tune-in toolkit: An open-source library for real-time binaural spatialisation. *PLoS ONE*, 14(3), March 2019. doi: doi:10.1371/journal.pone.0211899. URL https://doi.org/10.1371/journal.pone.0211899. 11

Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud computing: Issues and challenges. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 27–33, 2010. doi: 10.1109/aina.2010.187. 9

J. Dirksen. *Learn Three. Js: Program 3D Animations and Visualizations for the Web with JavaScript and WebGL*. Packt Publishing, Limited, 2023. ISBN 9781803233871. URL https://books.google.co.uk/books?id=J00_zwEACAAJ. 27

Boni García, Micael Gallego, Francisco Gortázar, and Antonia Bertolino. Understanding and estimating quality of experience in webrtc applications. *Computing*, 101(11):1585–1607, 2019. ISSN 1436-5057. doi: 10.1007/s00607-018-0669-7. URL https://doi.org/10.1007/s00607-018-0669-7. 11

Michael A Gerzon. Periphony: With-height sound reproduction. *Journal of the audio engineering society*, 21(1):2–10, 1973. URL http://www.aes.org/e-lib/browse.cfm?elib=2012. 10

Maksim Goman. How to improve risk management in it frameworks. In *2021 62nd International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)*, pages 1–6, 2021. doi: 10.1109/itms52826.2021.9615327. 12

Marcin Gorzel, Andrew Allen, Ian Kelly, Julius Kammerl, Alper Gungormusler, Hengchin Yeh, and Francis Boland. Efficient encoding and decoding of binaural sound with resonance audio. In *Audio Engineering Society Conference: 2019 AES International Conference on Immersive and Interactive Audio*, March 2019. URL http://www.aes.org/e-lib/browse.cfm?elib=20446. 11

Akio Honda, Hiroshi Shibata, Jiro Gyoba, Kouji Saitou, Yukio Iwaya, and Yôiti Suzuki. Transfer effects on sound localization performances from playing a virtual three-dimensional auditory game. *Applied Acoustics*, 68(8):885–896, 2007. ISSN 0003-682x. doi: https://doi.org/10.1016/j.apacoust.2006.08.007. URL https://www.sciencedirect.com/science/article/pii/S0003682X06001824. Head- Related Transfer Function and its Applications. 11

Jooyoung Hong, Jianjun He, Bhan Lam, Rishabh Gupta, and Woon-Seng Gan. Spatial audio for soundscape design: Recording and reproduction. *Applied Sciences*, 7:627, 06 2017. doi: 10.3390/app7060627. 11

William Hudson. User stories don't help users: introducing persona stories. *interactions*, 20 (6):50–53, 2013. 22

IT Governance Privacy Team. *Powers of Supervisory Authorities*. IT Governance Publishing, 2019. ISBN 9781787781924. URL https://app.knovel.com/hotlink/khtml/id:kt012G0L03/eu-general-data-protection/powers-supervisory-authorities. 16

Microsoft. Azure application architecture fundamentals, 08 2022. URL https://learn.microsoft.com/en-us/azure/architecture/guide/. 9

C. Minnick. *Beginning ReactJS Foundations Building User Interfaces with ReactJS: An Approachable Guide*. Wiley, 2022. ISBN 9781119685586. URL https://books.google.co.uk/books?id=J89cEAAAQBAJ. 11

Valerio Morucci. Reconsidering "cori spezzati": A new source from central italy. *Acta Musicologica*, 85(1):21–41, 2013. ISSN 00016241. URL http://www.jstor.org/stable/24595484. 10

Mozilla. Web audio api, 05 2019. URL https://developer.mozilla.org/en-US/docs/Web/API/Web%5FAudio%5FAPI. 11

Anna P. Murray. *The Complete Software Project Manager: Matering Technology from Planning to Launch and Beyond*. John Wiley & Sons, Incorporated, 2016. URL https://ebookcentral.proquest.com/lib/gmul-ebooks/detail.action?docID=4383484. 18

Markus Noisternig, Alois Sontacchi, Thomas Musil, and Robert Höldrich. A 3d ambisonic based binaural sound reproduction system. *Advances in Engineering Software*, January 2012. 10

Stephan Paul. Binaural recording technology: A historical review and possible future developments. *Acta Acustica united with Acustica*, 95:767–788, 09 2009. doi: 10.3813/aaa.918208. 10

Johan Pauwels. pywebaudioplayer: Bridging the gap between audio processing code in python and attractive visualisations based on web technology. In *4th Web Audio Conference (WAC)*, October 2018. URL https://qmro.qmul.ac.uk/xmlui/handle/123456789/60834. 11

Jenna Phipps. Asana vs clickup: Compare project management software. *CIO Insight*, page N.pag, 2022. ISSN 15350096. URL http://ezproxy.library.qmul.ac.uk/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=157666126&site=eds-live. 17

Archontis Politis, Symeon Delikaris-Manias, and Ville Pulkki. *Overview of Time–Frequency Domain Parametric Spatial Audio Techniques*. John Wiley & Sons, Ltd, 2017. ISBN 9781119252634. doi: https://doi.org/10.1002/9781119252634.ch4. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119252634.ch4. 10

C. Potts, K. Takahashi, and A.I. Anton. Inquiry-based requirements analysis. *Software, IEEE*, 11(2):21–32, 1994. 20

Project Management Institute. *A guide to the Project Management Body of Knowledge (PMBOK guide) and the standard for Project Management*. Project Management Institute, 2021. 12

Ling Qian, Zhiguo Luo, Yujian Du, and Leitao Guo. Cloud computing: An overview. In Martin Gilje Jaatun, Gansen Zhao, and Chunming Rong, editors, *Cloud Computing*, pages 626–631, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-10665-1. 6, 9

A. Roginska and P. Geluso. *Immersive Sound: The Art and Science of Binaural and Multi-Channel Audio*. Audio Engineering Society Presents. Taylor & Francis, 2017. ISBN 9781317480105. URL https://books.google.co.uk/books?id=elk6DwAAQBAJ. 10, 11

Rishi Shukla, Rebecca Stewart, Agnieszka Roginska, and Mark Sandler. User selection of optimal hrtf sets via holistic comparative evaluation. In *Audio Engineering Society Conference: 2018 AES International Conference on Audio for Virtual and Augmented Reality*, August 2018. URL http://www.aes.org/e-lib/browse.cfm?elib=19677. 11

Y. Suzuki, D. Brungart, K. Iida, D.A. Cabrera, Y. Iwaya, and H. Kato. *Principles And Applications Of Spatial Hearing*. World Scientific Publishing Company, 2011. ISBN 9789814465410. URL https://books.google.co.uk/books?id=N7rFCgAAQBAJ. 10, 11

Alp Ünver, Bita Kheibari, and Müge Sayıt. A webrtc architecture assisted by software defined networks. In *2020 28th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4, 2020. doi: 10.1109/siu49456.2020.9302042. 11

W3c. Web audio api, 06 2021. URL https://www.w3.org/TR/webaudio/. 11

Nicholas J. Wade and Hiroshi Ono. From dichoptic to dichotic: Historical contrasts between binocular vision and binaural hearing. *Perception*, 34(6):645–668, 2005. ISSN 0301-0066. URL http://ezproxy.library.qmul.ac.uk/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=psyh&AN=2005-09528-002&site=eds-live. 9

Karl Wiegers and Candase Hokanson. *Software Requirements Essentials: Core Practices for Successful Business Analysis*. Addison-Wesley, 2023. URL https://learning.oreilly.com/library/view/software-requirements-essentials/9780138190279/. 20, 21

K.E. Wiegers. Karl wiegers describes 10 requirements traps to avoid. *Software Testing & Quality Engineering*, 2(1), 2000. 20

B. Xie, R.P.I. Dr. Ning Xiang, and J. Blauert. *Head-Related Transfer Function and Virtual Auditory Display: Second Edition*. A Title in J. Ross Publishing's Acoustics: Information and Communication Series. J. Ross Publishing, 2013. ISBN 9781604270709. URL https://books.google.co.uk/books?id=fvDLCgAAQBAJ. 11

G. Zhai, J. Zhou, H. Yang, P. An, and X. Yang. *Digital TV and Wireless Multimedia Communications: 18th International Forum, IFTC 2021, Shanghai, China, December 3–4, 2021, Revised Selected Papers*. Communications in Computer and Information Science. Springer Singapore, 2022. ISBN 9789811922664. URL https://books.google.co.uk/books?id=fm1rEAAAQBAJ. 11