Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Spring 2024

Bits Please
OurScene

Harrison Cawood
Bryan Cox
Shawn Hyder
Vincent Nguyen
Daniel Palma
Sanny Tesfay

Revision History

Revision	Date	Author(s)	Description
1.0	3.27.2024	HC, BC, SH, VN, DP, ST	document creation

CONTENTS

List of Figures

List of Tables

1 Introduction

"Our Scene" intends to streamline the local music industry by implementing a central hub for each user in the process. The main feature will be a unified calendar system with different views based on the user specification (Fan, Artist, Promoter, and Venue), allowing all members to communicate schedules and view when time slots are booked or available. The fans will be able to see when their favorite bands are performing as well as new music at known or unknown venues. The artist will be able to see what time slots are being shared by the promoter and also see what other artists are performing in the same set. The promoter will be able to schedule artists with ease allowing them to market to unknown artists and work closely with the venue. The venue will be able to see all that is happening in the scheduling process and make decisions if need be. After a gig is booked, a channel is created so that fans of one artist can see who else is performing and possibly discover a new artist to start following as well as giving a preview into what the show will be like. To assist in the promoterâs search for artists as well as the artistâs attempt to be discovered, the artist will be able to upload their music through Spotify onto the site, which will be viewable by a fan trying to find new music. On top of being able to hear the artistâs music, fans can vote and boost songs or artists they enjoy helping get their music discovered and booked more often.

2 System Overview

There are three layers to *Our Scene*; the front-end, CRUD Layer, and API. The front-end consists of what the user will interact with to navigate the application. The back-end will be where various items such as songs or user data will be stored. The API will be how the front-end, back-end, and other connected applications will interact and communicate with each other.

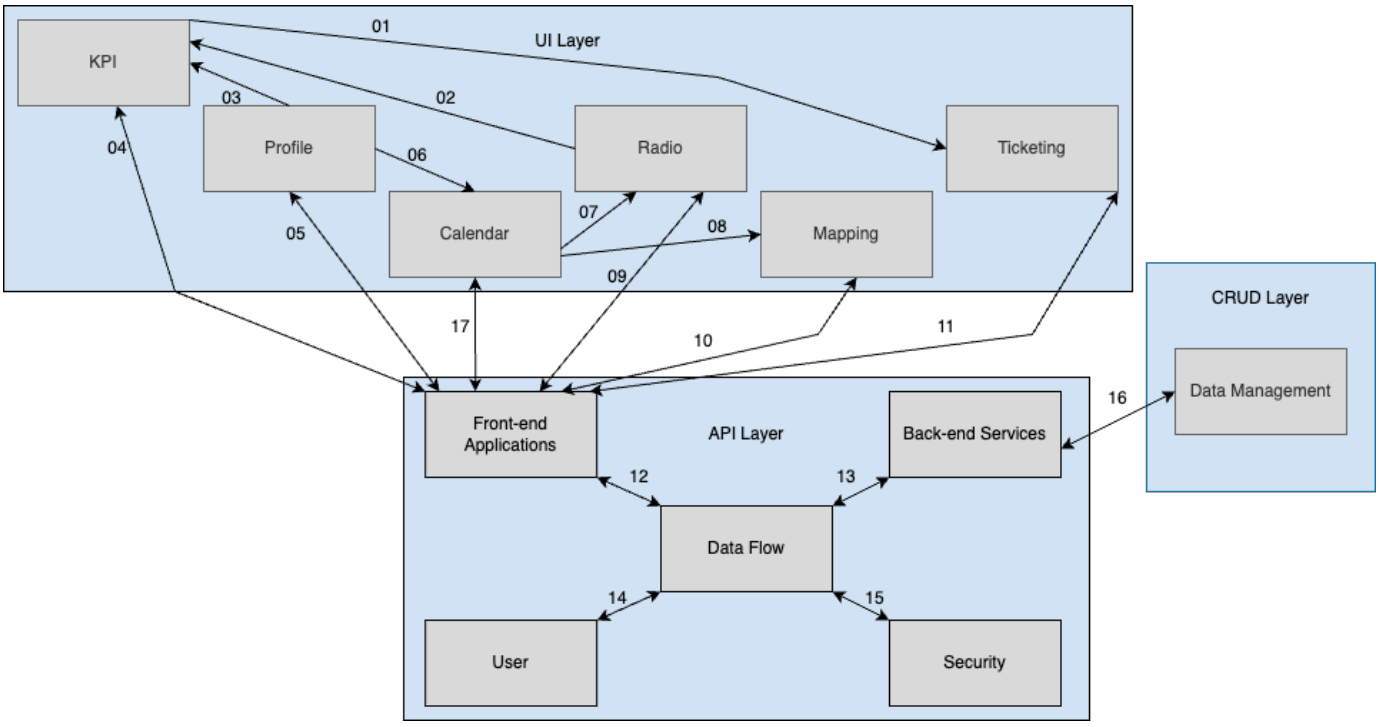

Figure 1: System Architecture

3 UI LAYER SUBSYSTEMS

UI Layer Design Overview: This section outlines the software design of the UI layer, focusing on the specific frontend technologies utilized to enable user expression. It facilitates users in creating personalized pages to express themselves and provide contact information. Additionally, users can explore other profiles and connect with each other. Essentially, this layer constitutes a crucial component of the software application, enabling interaction between users and the system, while also serving as a conduit for converting application data changes into a format suitable for presentation and display within the UI.

3.1 LAYER HARDWARE

The UI Layer uses technologies specifically for front-end, therefore there are no utilization of hardware.

3.2 LAYER OPERATING SYSTEM

Operating systems are not required for the UI Layer.

3.3 LAYER SOFTWARE DEPENDENCIES

- Next.js: Supplies a React framework that furnishes the foundational elements for crafting web applications. It manages the requisite tools and setup for React while offering enhanced organization, functionalities, and performance enhancements tailored for our specific application needs.
- Tailwind CSS: A CSS framework characterized by a utility-first approach, which simplifies web development through a collection of pre-designed utility classes.
- Shadcn: A collection of meticulously designed, accessible, and adaptable React components ideal for constructing modern web applications using Next.js.
- Zod: A TypeScript schema declaration and validation library enabling the design of schemas for various data types, including primitives, objects, arrays, and more.
- Lucide: An open-source icon library offering over 1000 vector (SVG) files, serving as a comprehensive resource for displaying icons and symbols across digital and non-digital projects.

3.4 SUBSYSTEM 1: PROFILE

This subsystem allows users to create their own page that contains information about the user. It also allows users to view other users' profiles and connect with each other.

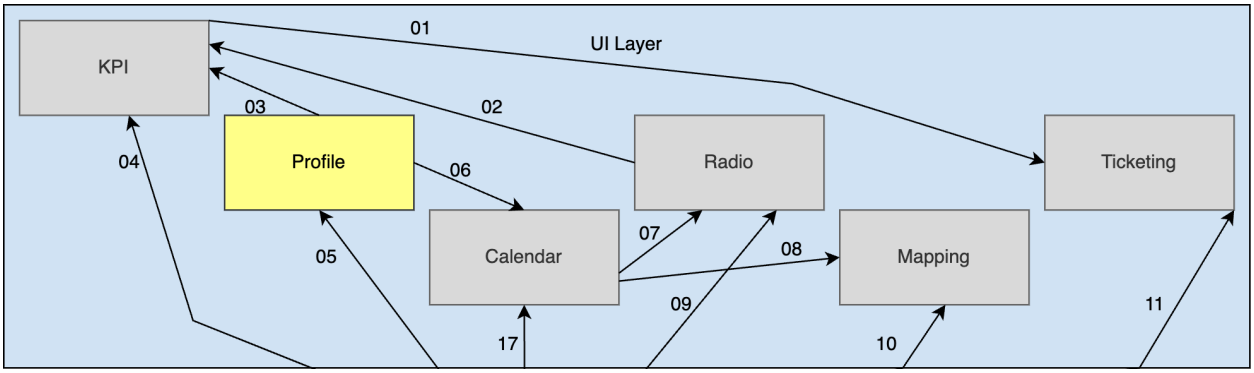

Figure 2: UI layer Profile subsystem

3.4.1 Subsystem Software Dependencies

- Next.js
- Tailwind CSS
- Shadcn
- Zod
- Lucide

3.4.2 Subsystem Programming Languages

- HTML: a markup language used to structure content on the web, defining the layout and elements of a webpage.
- CSS: a stylesheet language used to describe the presentation of HTML and XML documents, enhancing their appearance and layout.
- JavaScript: a versatile programming language primarily used for adding interactivity and dynamic behavior to web pages, enabling client-side scripting and interaction with web browsers.
- TypeScript: a superset of JavaScript that adds optional static typing and other advanced features, enhancing developer productivity and facilitating the development of large-scale JavaScript applications.

3.4.3 Subsystem Data Structures

Given the distinct focus of this layer on the frontend aspect of the application, with data structure considerations primarily pertinent to the backend, it is unnecessary to elaborate on the data structure within this layer.

3.4.4 Subsystem Data Processing

As this layer is dedicated to the frontend of the application, while data processing concerns are typically handled by the backend, a detailed description of the layer's data structure is deemed unnecessary. However, it's worth noting that this subsection interfaces with the backend to retrieve relevant data.

3.5 Subsystem 2: Calendar

The calendar allows users to view, arrange, or book upcoming events.

3.5.1 Subsystem Software Dependencies

- Next.js
- Tailwind CSS
- Shadcn
- Zod
- Lucide

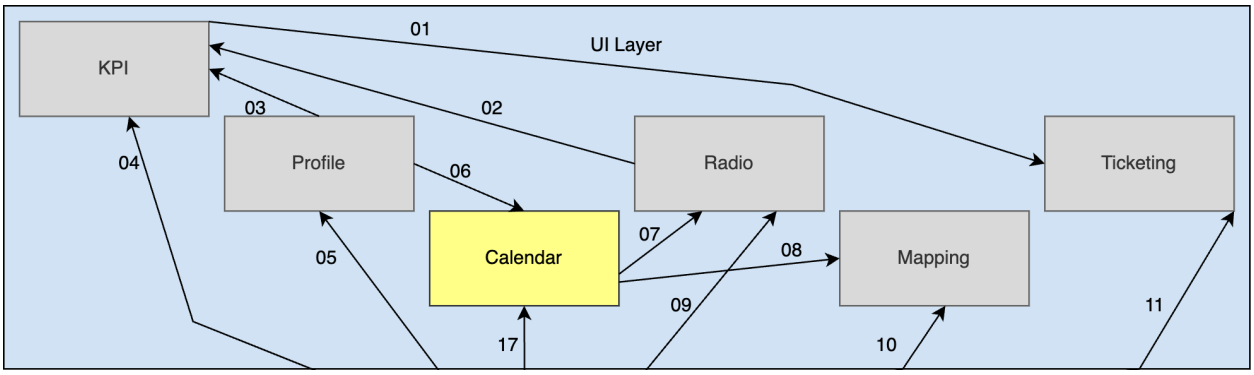

Figure 3: UI layer Calendar subsystem

3.5.2 Subsystem Programming Languages

- HTML: a markup language used to structure content on the web, defining the layout and elements of a webpage.
- CSS: a stylesheet language used to describe the presentation of HTML and XML documents, enhancing their appearance and layout.
- JavaScript: a versatile programming language primarily used for adding interactivity and dynamic behavior to web pages, enabling client-side scripting and interaction with web browsers.
- TypeScript: a superset of JavaScript that adds optional static typing and other advanced features, enhancing developer productivity and facilitating the development of large-scale JavaScript applications.

3.5.3 Subsystem Data Structures

Given the distinct focus of this layer on the frontend aspect of the application, with data structure considerations primarily pertinent to the backend, it is unnecessary to elaborate on the data structure within this layer.

3.5.4 Subsystem Data Processing

As this layer is dedicated to the frontend of the application, while data processing concerns are typically handled by the backend, a detailed description of the layer's data structure is deemed unnecessary. However, it's worth noting that this subsection interfaces with the backend to retrieve relevant data.

3.6 Subsystem 3: KPI

Key Performance Indicator (KPI) provides a measurement of performance over time for objectives. This allows vendors, promoters, and bands to view their progress and help them choose their next steps.

3.6.1 Subsystem Software Dependencies

- Next.js
- Tailwind CSS
- Shadcn
- Zod
- Lucide

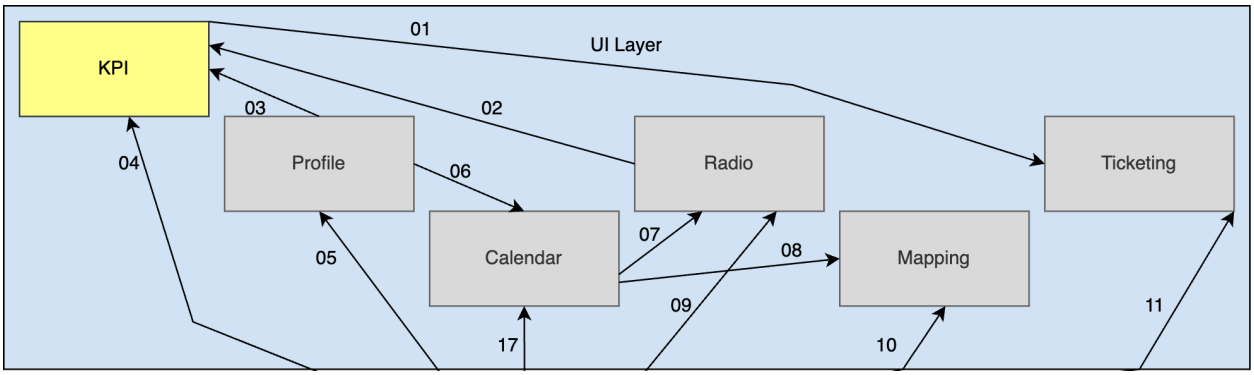

Figure 4: UI layer KPI subsystem

3.6.2 Subsystem Programming Languages

- HTML: a markup language used to structure content on the web, defining the layout and elements of a webpage.
- CSS: a stylesheet language used to describe the presentation of HTML and XML documents, enhancing their appearance and layout.
- JavaScript: a versatile programming language primarily used for adding interactivity and dynamic behavior to web pages, enabling client-side scripting and interaction with web browsers.
- TypeScript: a superset of JavaScript that adds optional static typing and other advanced features, enhancing developer productivity and facilitating the development of large-scale JavaScript applications.

3.6.3 Subsystem Data Structures

Given the distinct focus of this layer on the frontend aspect of the application, with data structure considerations primarily pertinent to the backend, it is unnecessary to elaborate on the data structure within this layer.

3.6.4 Subsystem Data Processing

As this layer is dedicated to the frontend of the application, while data processing concerns are typically handled by the backend, a detailed description of the layer's data structure is deemed unnecessary. However, it's worth noting that this subsection interfaces with the backend to retrieve relevant data.

3.7 Subsystem 4: Radio

This subsystem allows users to explore and listen to bands and artists.

3.7.1 Subsystem Software Dependencies

- Next.js
- Tailwind CSS
- Shadcn
- Zod
- Lucide

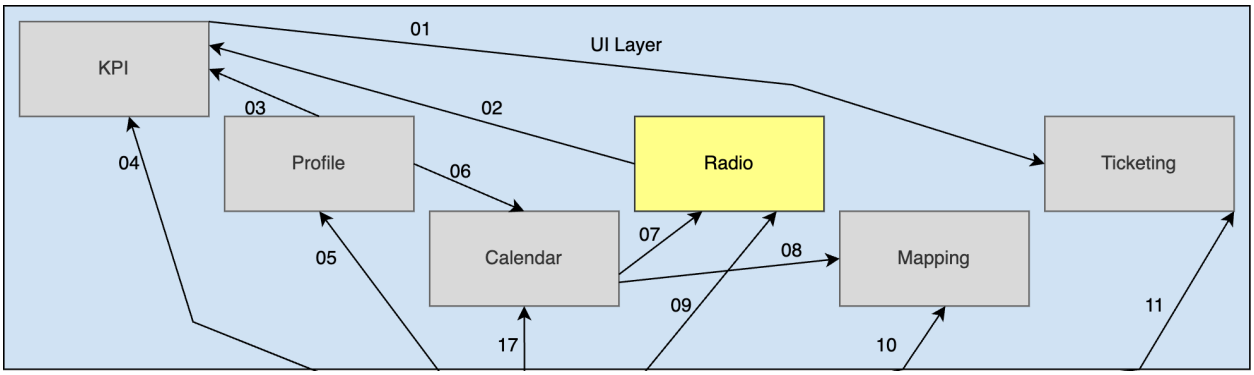

Figure 5: UI layer Radio subsystem

3.7.2 Subsystem Programming Languages

- HTML: a markup language used to structure content on the web, defining the layout and elements of a webpage.
- CSS: a stylesheet language used to describe the presentation of HTML and XML documents, enhancing their appearance and layout.
- JavaScript: a versatile programming language primarily used for adding interactivity and dynamic behavior to web pages, enabling client-side scripting and interaction with web browsers.
- TypeScript: a superset of JavaScript that adds optional static typing and other advanced features, enhancing developer productivity and facilitating the development of large-scale JavaScript applications.

3.7.3 Subsystem Data Structures

Given the distinct focus of this layer on the frontend aspect of the application, with data structure considerations primarily pertinent to the backend, it is unnecessary to elaborate on the data structure within this layer.

3.7.4 Subsystem Data Processing

As this layer is dedicated to the frontend of the application, while data processing concerns are typically handled by the backend, a detailed description of the layer's data structure is deemed unnecessary. However, it's worth noting that this subsection interfaces with the backend to retrieve relevant data.

3.8 Subsystem 5: Ticketing

This subsystem allows the booking of concerts to take place, as well as the exchange of information of the events.

3.8.1 Subsystem Software Dependencies

- Next.js
- Tailwind CSS
- Shadcn
- Zod
- Lucide

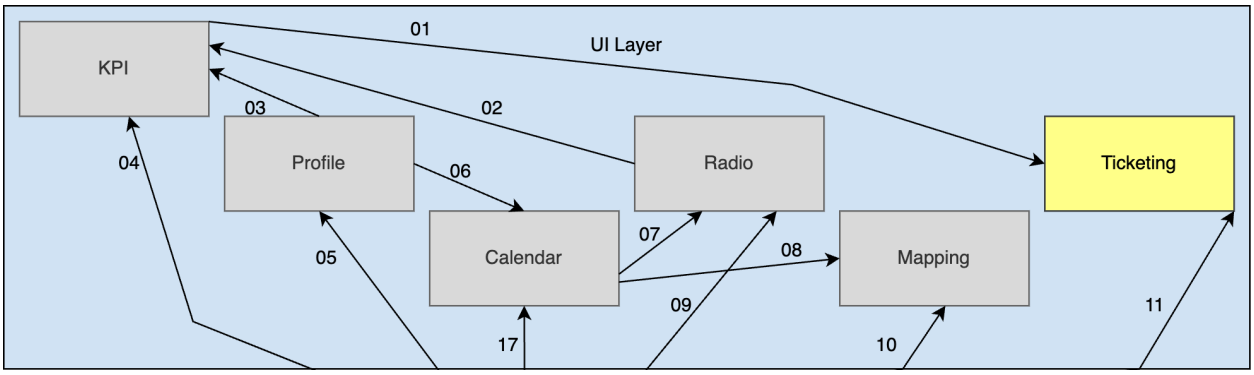

Figure 6: UI layer Ticketing subsystem

3.8.2 Subsystem Programming Languages

- HTML: a markup language used to structure content on the web, defining the layout and elements of a webpage.
- CSS: a stylesheet language used to describe the presentation of HTML and XML documents, enhancing their appearance and layout.
- JavaScript: a versatile programming language primarily used for adding interactivity and dynamic behavior to web pages, enabling client-side scripting and interaction with web browsers.
- TypeScript: a superset of JavaScript that adds optional static typing and other advanced features, enhancing developer productivity and facilitating the development of large-scale JavaScript applications.

3.8.3 Subsystem Data Structures

Given the distinct focus of this layer on the frontend aspect of the application, with data structure considerations primarily pertinent to the backend, it is unnecessary to elaborate on the data structure within this layer.

3.8.4 Subsystem Data Processing

As this layer is dedicated to the frontend of the application, while data processing concerns are typically handled by the backend, a detailed description of the layer's data structure is deemed unnecessary. However, it's worth noting that this subsection interfaces with the backend to retrieve relevant data.

3.9 Subsystem 6: Mapping

This subsystem provides the location information of where the event is being held.

3.9.1 Subsystem Software Dependencies

- Next.js
- Tailwind CSS
- Shadcn
- Zod
- Lucide

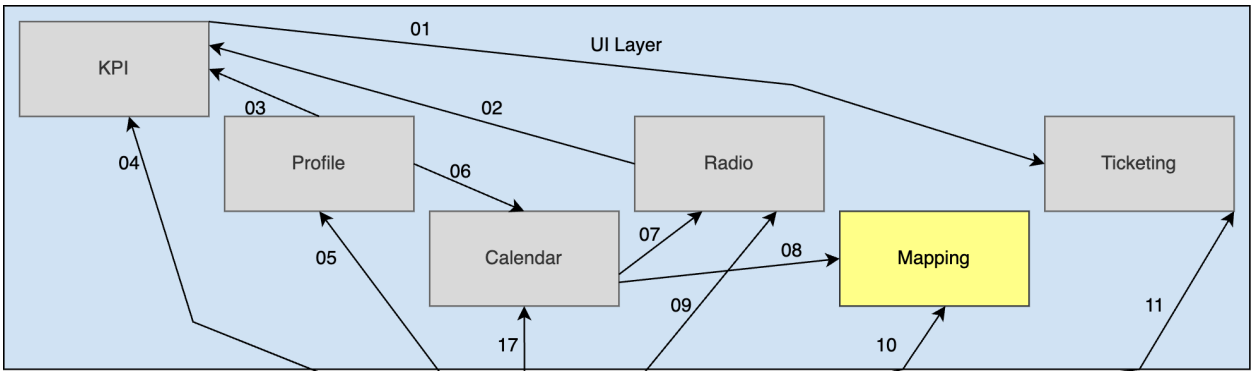

Figure 7: UI layer Mapping subsystem

3.9.2 Subsystem Programming Languages

- HTML: a markup language used to structure content on the web, defining the layout and elements of a webpage.
- CSS: a stylesheet language used to describe the presentation of HTML and XML documents, enhancing their appearance and layout.
- JavaScript: a versatile programming language primarily used for adding interactivity and dynamic behavior to web pages, enabling client-side scripting and interaction with web browsers.
- TypeScript: a superset of JavaScript that adds optional static typing and other advanced features, enhancing developer productivity and facilitating the development of large-scale JavaScript applications.

3.9.3 Subsystem Data Structures

Given the distinct focus of this layer on the frontend aspect of the application, with data structure considerations primarily pertinent to the backend, it is unnecessary to elaborate on the data structure within this layer.

3.9.4 Subsystem Data Processing

As this layer is dedicated to the frontend of the application, while data processing concerns are typically handled by the backend, a detailed description of the layer's data structure is deemed unnecessary. However, it's worth noting that this subsection interfaces with the backend to retrieve relevant data.

4 CRUD Layer subsystems

CRUD Layer Design - This section provides an exhaustive outline of the CRUD layer's hardware and software design, focusing on the specific technologies employed to facilitate Create, Read, Update, and Delete operations within the system. Given the project's reliance on Amazon Web Services (AWS) products, this document delineates how these services integrate to form a cohesive backend infrastructure.

4.1 Layer Hardware

The CRUD layer is primarily software-centric with no direct hardware components. It operates on cloud infrastructure provided by AWS, negating the need for dedicated hardware discussion in this context.

4.2 Layer Operating System

The CRUD layer is agnostic to the operating system due to its deployment on AWS, which abstracts the underlying OS. Developers interact with AWS services without managing OS-level configurations directly.

4.3 Layer Software Dependencies

- AWS Amplify: Provides a framework for connecting the backend with frontend applications, facilitating easier access to other AWS services like Amazon Cognito, AWS AppSync, and Amazon DynamoDB.
- AWS AppSync: A managed GraphQL service that simplifies data synchronization and manipulation across scalable applications, leveraging GraphQL's capabilities for efficient data retrieval.
- Amazon Cognito: Manages user authentication and authorization, seamlessly integrating with other AWS services to secure backend access.
- Amazon DynamoDB: A NoSQL database service offering fast and predictable performance with seamless scalability for storing and retrieving any amount of data.
- Amazon S3: Provides object storage through a web service interface, used here for storing static resources.

4.4 Subsystem 1: Data Management

The primary aim of the Data Management subsystem is to ensure robust, secure, and efficient handling of data. It serves as the backbone for data transactions between the user interfaces and the database storage, facilitating:

- User authentication and authorization
- Data storage and retrieval
- Data synchronization
- Real-time data updates

4.4.1 Subsystem Hardware

Not applicable, as this subsystem operates entirely within AWS's managed services.

4.4.2 Subsystem Operating System

Not applicable, as AWS services abstract away the need for direct OS management.

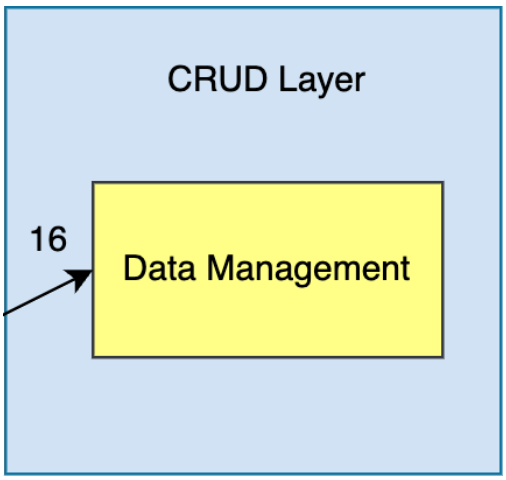

Figure 8: CRUD layer Data Management subsystem

4.4.3 Subsystem Software Dependencies

- AWS Amplify
- AWS AppSync
- Amazon Cognito
- Amazon DynamoDB
- Amazon S3

These services collectively handle the data management needs of the application, from authentication to data storage and retrieval.

4.4.4 Subsystem Programming Languages

- GraphQL: Used with AWS AppSync for efficient data operations.
- AWS SDKs: Depending on the application's frontend technology, AWS SDKs (e.g., JavaScript, Python) are used to interact with AWS services.

4.4.5 Subsystem Data Structures

Data is structured according to the requirements of DynamoDB tables and S3 object storage conventions. For GraphQL operations, data is defined by schemas that represent the types and relationships within the application's data model.

4.4.6 Subsystem Data Processing

Data processing involves CRUD operations facilitated by AWS AppSync and DynamoDB, leveraging GraphQL for efficient data fetching and manipulation. Security and access control are managed through Amazon Cognito, ensuring that data operations are authenticated and authorized.

This design leverages AWS's robust ecosystem to create a secure, scalable, and efficient CRUD layer, focusing on seamless integration and abstraction to ease development efforts and ensure a scalable application architecture.

5 API Subsystems

API Layer design overview: This section describes the design and software needed to implement the API Layer. The API layer regards the application communication interfaces and how different parts of the application will communicate, update and Transfer information to each other.

5.1 Layer Hardware

The Project is a software application, with its reliance on AWS Services this means that hardware will not be discussed.

5.2 Layer Operating System

OurScene is an AWS distributed software application for web devices, therefore the application is operating system agnostic.

5.3 Layer Software Dependencies

- AWS Amplify: Amplify will provide a means to connect the front-end application with the back-end services holding it up. Amplify simplifies the interaction of the AWS back-end services via API endpoints. Lastly Amplify provides an abstraction to simplify the Synchronization, Authentication and storage of data from AWS services.
- AWS AppSync: AppSync will play a central role in this layer. The interface will allow for us powerful definition and Implementation of the needed graphQL APIs, which allow for data operations using queries, mutations and subscriptions. Real-time data updates and optimal performance of the API Layer are provided by AppSync.
- AWS Cognito: Cognito provides the API Layer with management of User Authentication as well as authorization making it a critical component for the layer. The service offers the API Layer a set of features for securing API endpoints, User ID management and implementing the workflow of User AUthentication. Finally, in this layer, Cognito will be used for API Request authentication, access control and user session management.
- Amazon DynamoDB: Serving as the primary data storage solution for the back-end services, DynamoDB is fundamental to the API Layer. Since DynamoDB is faster due to being a NoSQL database, it is perfect handling the various data management tasks in the layer.
- Amazon S3: S3 is essential for this layer, both storing and being able to access the resources stored in cases such as music for the radio, and KPI functions especially. S3 will allow the layer to store music mostly in this application as the rest will be stored in DynamoDB

5.4 Security subsystem

The Security subsystem will be responsible for Authenticating users, access control to resources and pages and application security.

5.4.1 Security subsystem Software Dependencies

- Passport.js for Node.js based authentication
- Amazon Cognito, will handle the application's security for the most part.

5.4.2 Security subsystem Programming Languages

- JavaScript Node.js for backed implementaton

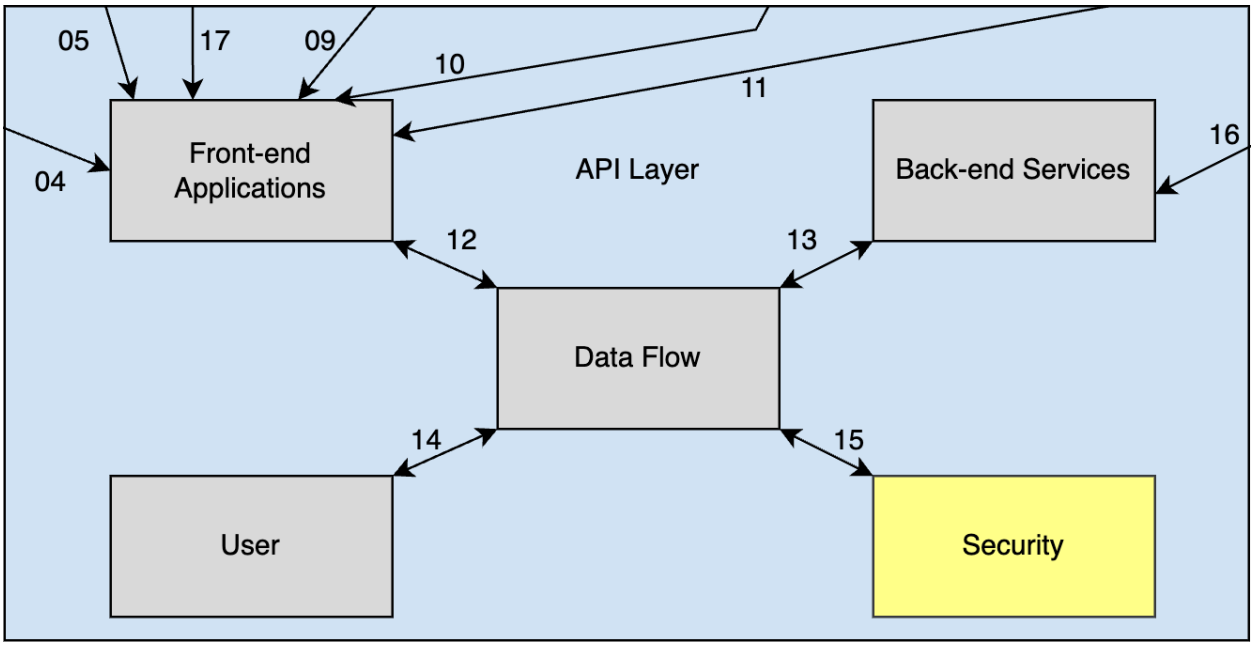

Figure 9: API layer Security subsystem

5.4.3 Security subsystem Data Structures

- User data model stored in DynamoDB or Amplify user pool
- JWT tokens for AUth

5.4.4 Security subsystem Data Processing

- AWS Amplify
- Amazon Cognito

5.5 User subsystem

The User subsystem is in charge of user managagement. It will handle user login, security, preferences and Data.

5.5.1 User subsystem Software Dependencies

- Amazon Cognito will be used for user authentication, authorization and ID management.
- AWS Amplify will make for an abstraction from AWS services for ease of use.
- AWS AppSync for database connectivity to DynamoDB
- React for Front end Libraries.

5.5.2 User subsystem Programming Languages

- Node.js for the back end
- HTML for front end
- CSS for front end
- JavaScript for front end

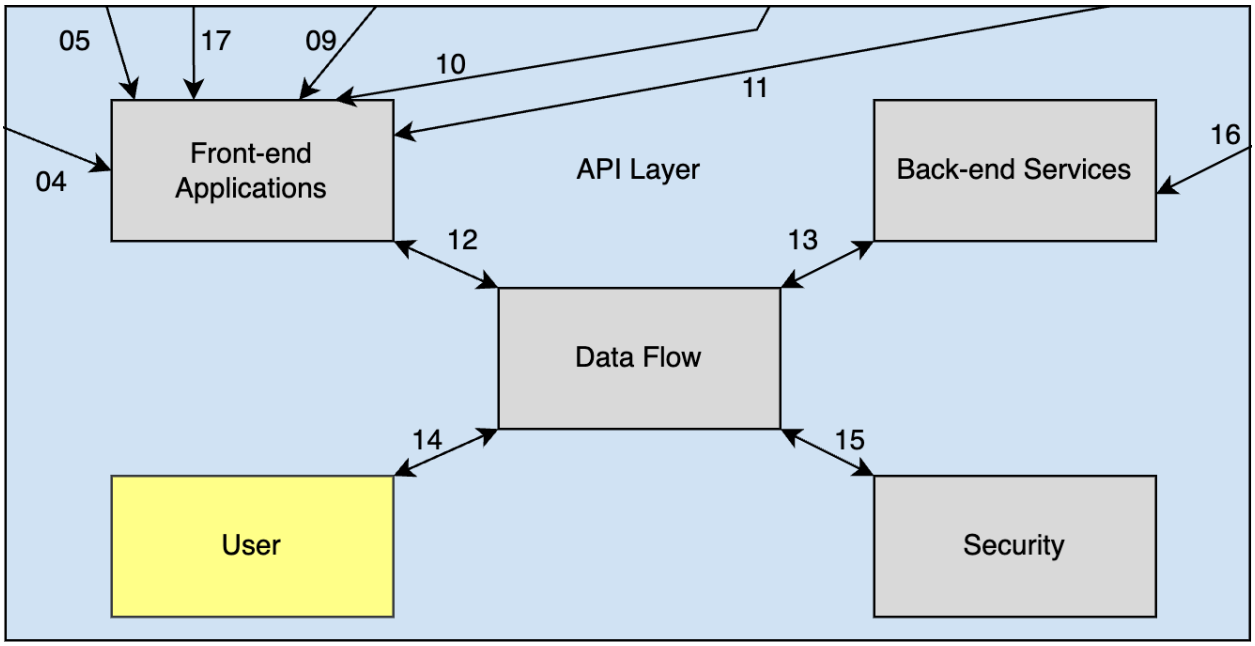

Figure 10: API layer User subsystem

5.5.3 User subsystem Data Structures

- Relationships between the Different Users in the DynamoDB tables

5.5.4 User subsystem Data Processing

- CRUD operations using the AWS Amplify API for user data
- Input validation and Sanitization for User Input

5.6 Data Flow subsystem

The Data Flow subsystem will be in charge of integrating all of the different backend services like databases, authentication, security etc. and delivering that data to the appropriate pages. This is mainly done by creating forms for how the APIs will accept and give data.

5.6.1 Data Flow subsystem Software Dependencies

- AWS Amplify
- Amazon AppSync
- Amazon DynamoDB
- Amazon S3

5.6.2 Data Flow subsystem Data Structures

- Forms for how the Various APIs will communicate data will need to be created.
- Tables in DynamoDB
- Music storage in Amazon's S3 storage Bucket

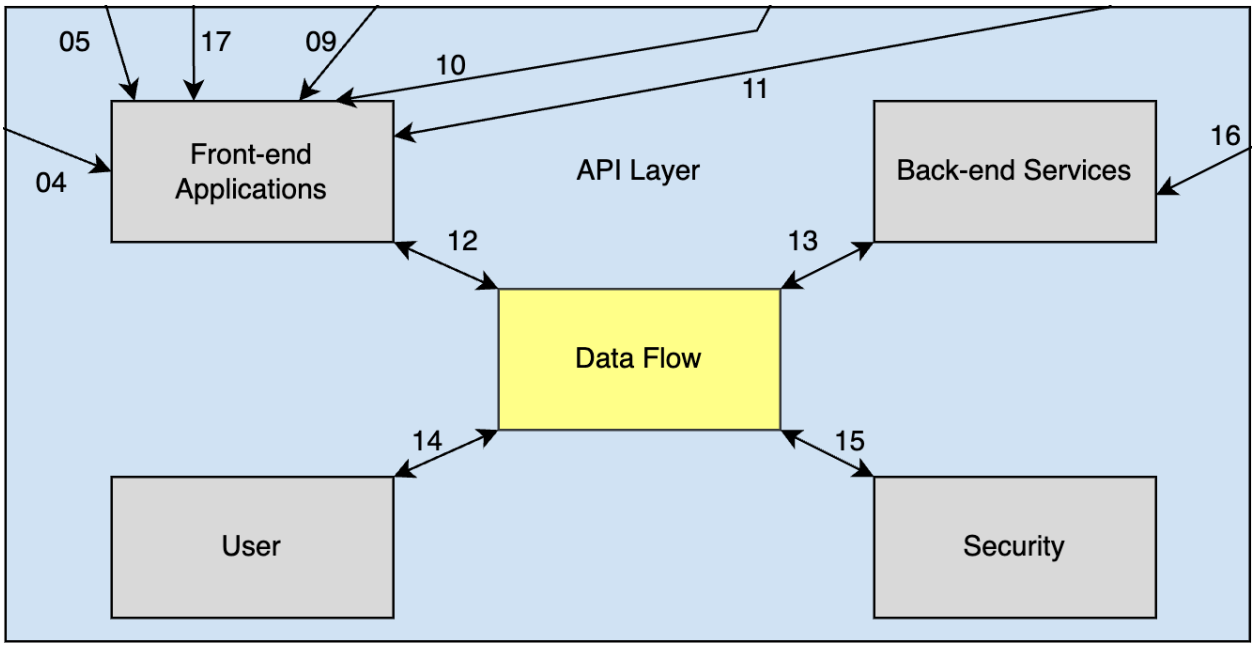

Figure 11: API layer Data Flow subsystem

5.7 Back End subsystem

The Back end subsystem will be used to update and synchronize the back end elements with what the users will do on the front end, as well as will use and perform operations on data to update the front end properly.

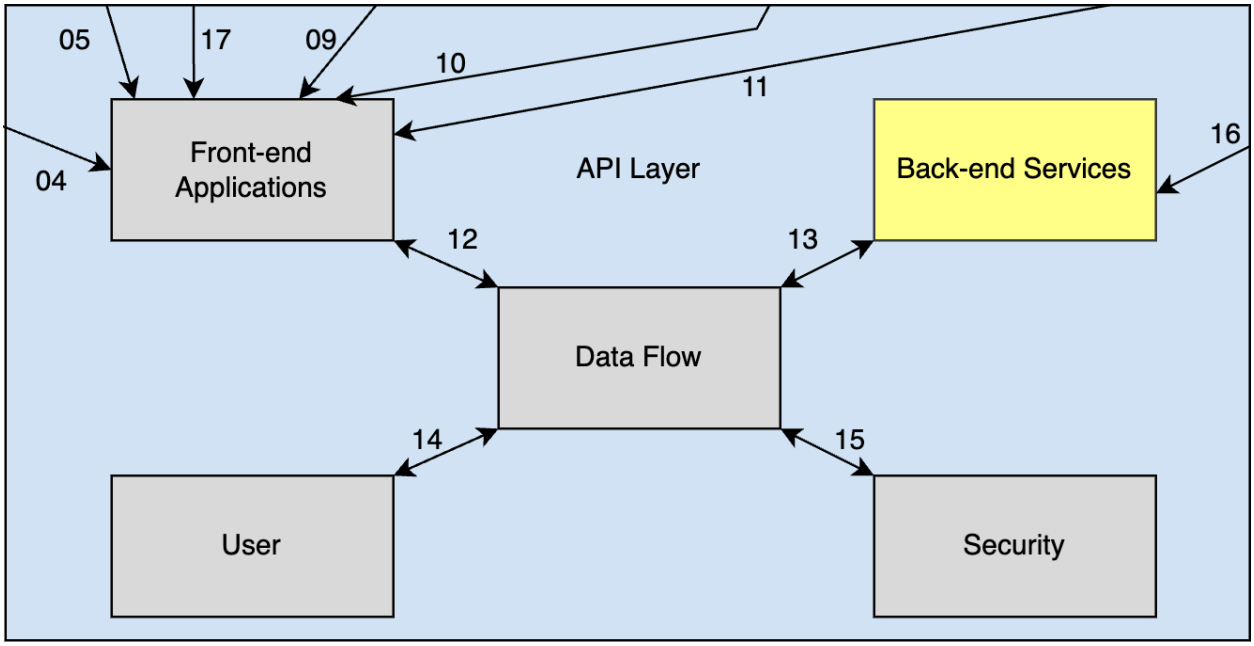

Figure 12: API layer Back-end subsystem

5.7.1 Back End subsystem Software Dependencies

- Express.js for node based backend dev
- AWS Amplify/AppSync for Database connectivity

5.7.2 Back End subsystem Programming Languages

- JavaScript Node.JS for backend programming

5.7.3 Back End Subsystem Data Structures

- Models for storing data in DynamoDB

5.7.4 Back End subsystem Data Processing

- CRUD calls on DynamoDB data
- Authorization and Authentication from Amazon Cognito

5.8 Front end subsystem

The Front end subsystem in the API layer will ensure how data will be updated in the presentation layer for each user since the UI will be different based on the type of the User.

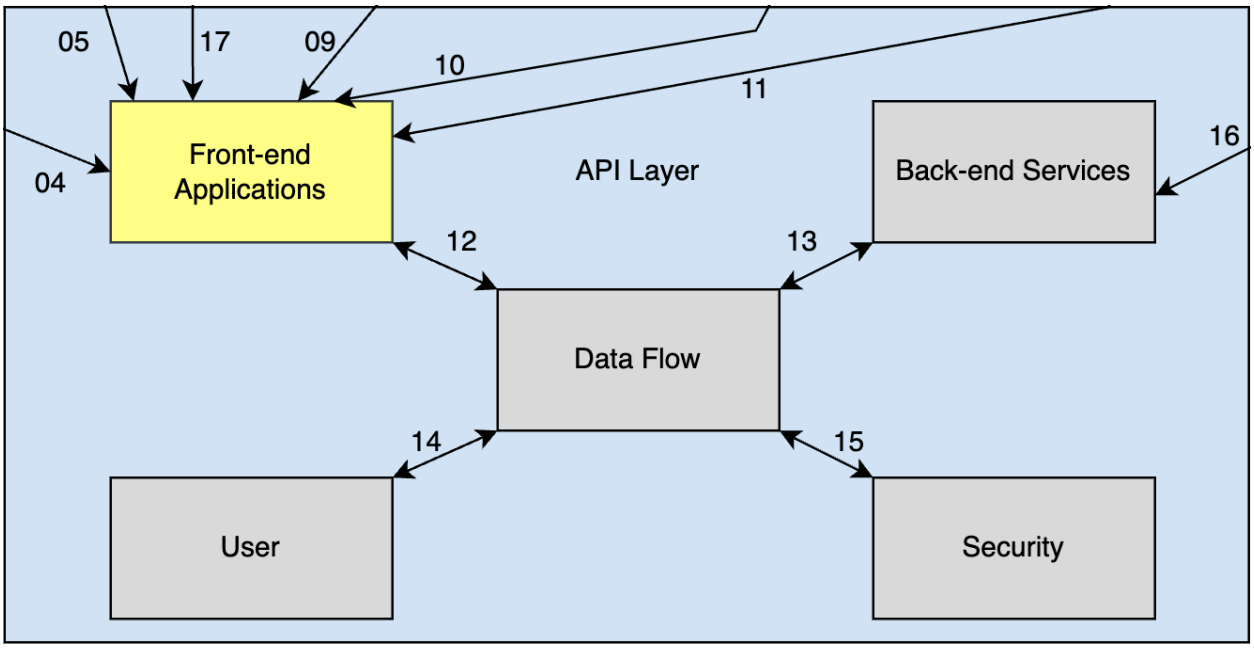

Figure 13: API layer Front-end Applications subsystem

5.8.1 Front End subsystem Software Dependencies

- React

5.8.2 Front End subsystem Programming Languages

- HTML
- CSS
- JavaScript
-

5.8.3 Front End subsystem Data Structures

- Components for UI organization in front end frameworks

5.8.4 Front End subsystem Data Processing

The data processing for this subsystem of the API Layer will involve using backend APIs and database calls.