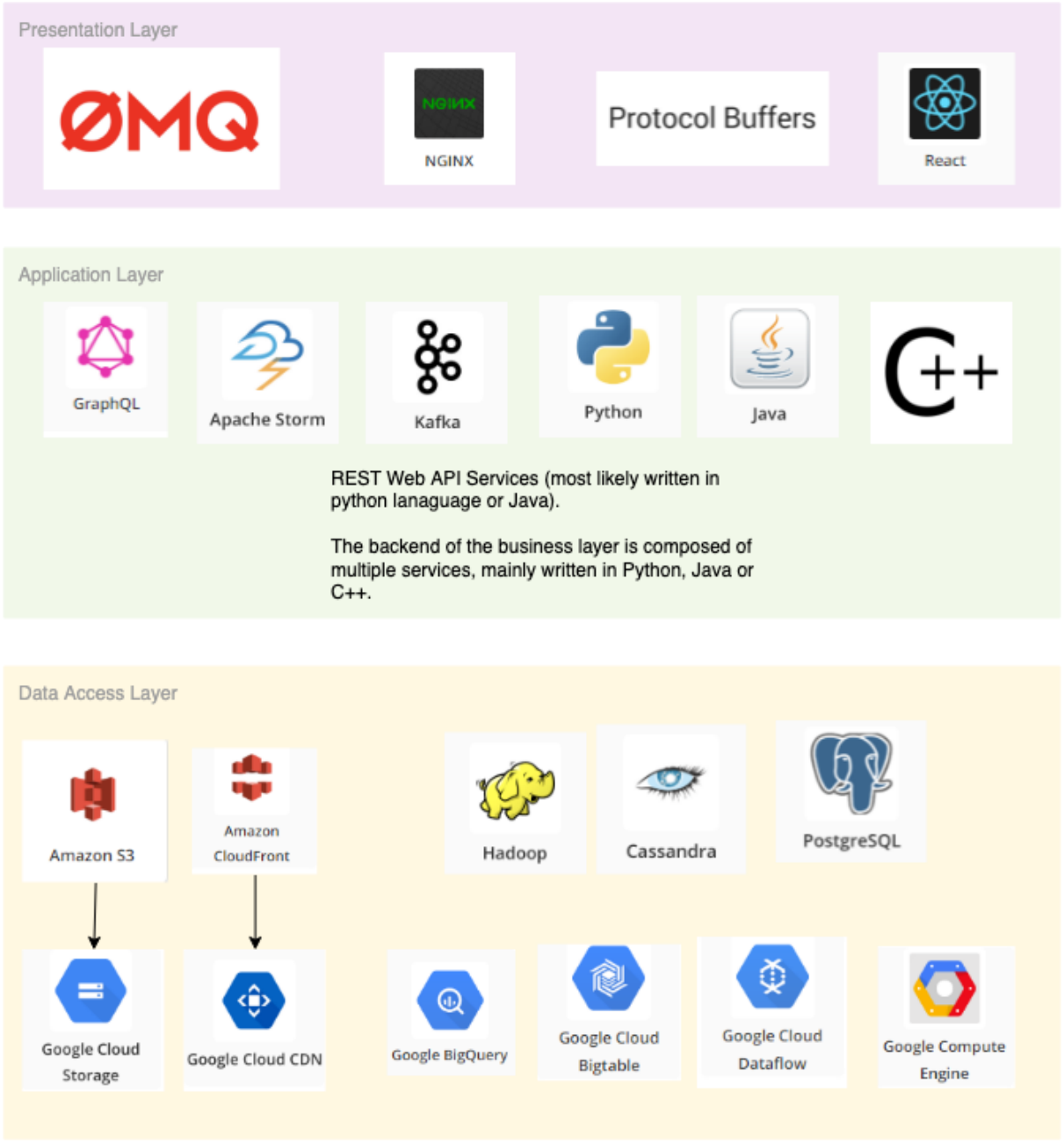
1. Presentation, Application and Data layer.

Spotify Web Player



Presentation Layer

1. This is where the software interacts directly with the user. Features such as the Home, Search, My Library and Settings can be accessed. A special feature of the app is utilising a different User Interface when used together with other navigational apps such as Google Maps or Waze. This is done on the client side also.

2. Spotify services in the backend are mainly written in Python and Java. These services communicate mainly on their own protocol built on the server libraries of ZeroMQ and ProtoBuf.

Application Layer

Apache Kafka is used to handle a big amount of data in fractions of a second. This is similar the millions of listeners on Spotify trying to listen to Justin Bieber's new song - Kafka allows Spotify to scale its streaming service to a larger audience.

Apache Storm ensures full data security and is used as a distributed framework or real-time computation. It takes data from data sources such as Kafka and Cassandra and processes the data in real time (written in Java).

GraphQL is used to execute queries on existing data. The technology falls into the application layer of Spotify's tech stack. It provides easy-to-understand and complete description of the data in the API. They are a fast and stable way to get data because they control the data and not the server itself.

Data Storage Layer

Storage is primarily PostgreSQL, Cassandra or Hadoop. These static indexes are primarily used for searching song titles/artists/podcasts or can also be used to store metadata such as User details, or payment details.

Audio files were stored on a cloud base service such as Amazon S3 and cached using Amazon Cloudfront as a content delivery network for low latency (techstack.io). However, since 2016, according to an article published by Alex Konrad on www.forbes.com, Spotify's main data has migrated onto the Google Cloud Platform. The Google Cloud storage is similar to Amazon S3 and Google Cloud CDN is similar to Amazon's Cloudfront. Other Google services that Spotify utilises is Google BigQuery which is a data analysis tool capable of running super quick, SQL-like queries through terabytes of data.

The overall advantages of this RESTful architecture promotes the scalability, performance and highly adaptable streaming services of Spotify.

2. Role of 3 technologies and how they relate to neighbouring technologies.

Spotify initially had two client systems for desktop UI - a Desktop Client application and a Web Player client. The Web Player uses the ReactJS architecture whilst at the time the Desktop client used a diverse range of web applications. Because they had two clients, it was very hard for the engineering team to ship new features to both the Desktop and Web Player client separately. In general, Spotify overcame this problem by implementing the ReactJS and GraphQL application for both clients.

GraphQL is used to execute queries on existing data. The technology falls into the application layer of Spotify's tech stack. It provides easy-to-understand and complete description of the data in the API. They are a fast and stable way to get data because they control the data and not the server itself.

ReactJS is used to make interactive web applications. It is a JavaScript library that creates interactive user interfaces. It can change data on a website page without having to refresh the page. This allows the web application to be faster, more scalable and simple.

The User Interface built on ReachJS communicates with Spotify's backend through GraphQL and other Web API services. From the backend, the clients can access audio files stored on most likely a database such as Google Cloud Storage and cached on Google Cloud CDN. The Cloud Storage service is a highly durable and scalable cloud storage solution that most likely stores the main database of Spotify's audio files. Cloud CDN essentially enables users to access these datas with greater speed and reliability because information is cached, allowing for faster load times.

3. Pick 2 technologies and discuss what physical hardware it is most likely delivered on.

When creating new buckets, any object/data uploaded on to Google Cloud Storage uses *storage classes* as a piece of metadata. A default storage class can be set for the bucket. The primary classes offered by Google Cloud Storage are Standard Storage, Nearline Storage, Coldline Storage, and Archive Storage. Spotify most likely is delivered on a Standard Storage because data gets frequently accessed. It is most likely co-located with Compute Engine instances that use the data. This enhances the performance and output for data-heavy computation and streaming service of what Spotify is.

Likewise, there are different machine type families for Google's Compute Engine.Virtual Machines are classed by machine family. Some include E2, N2, E2 Shared-core, N2D and Tau T2D. With Spotify's workload being mainly a media/streaming service, it most likely utilises the N2, N2D, or N1 General Purpose Balanced Virtual Machines. All machines have at least 6.5GB of memory per virtual central processing unit (vCPU) and offer up to 224 vCPUs. N2 VMs offer Intel Ice Lake and Cascade Lake CPU platforms, whilst N2D has the 2nd generation AMD EPYC Rome platforms. Finally, N1 offers Intel Sandy Bridge, Ivy Bridge, and Haswell among other CPU platforms. All VMs offer customizability, local SSDs and preemptible VMs.

4. Create table to compare features and benefits of similar alternative technology

	Google Cloud Storage	Amazon S3
How efficiently big data's get uploaded	In GCS, we would upload parts of an object as an individual object. Logic is then required as we need to implement a compose method on the list of "parts". The compose command is then repeated by batches until the entire object is combined together.	With Amazon S3, objects can be uploaded via multiple HTTP requests. After all parts have been uploaded, S3 constructs the object from all the uploaded parts

How they tag objects with simple browsable metadata to objects via key-value pairs	N/A	Implemented since 2016.
How well they retrieve objects that are deleted or overwritten	Removing the object without specifying the branch will move it from the main branch to an 'archive' branch without creating a DELETE MARKER. However, specifying the branch/version ID will permanently destroy the object/data without moving it to the archive.	AWS has a main branch that references the most recent entry. Any commits that do not specify a branch automatically change the main branch - including the DELETE operation. Deleting the object creates a DELETE MARKER. Can get the deleted/overwritten branch by using the commit/version ID.
How well the tech copies data across buckets in different locations. Do they keep a copy of your data as insurance?	By selecting a 'Multi-Regional Storage' bucket location, data can be copied across to the selected location of choice. Similar to the S3 however, the region is not a personalised choice.	S3 has two way sync of buckets and implements 'Cross Region Replication (CRR)'. Any object that gets uploaded to an S3 bucket will automatically be sent to a different destination bucket in a different region of our choice.

	Google Compute Engine	Amazon AWS EC2
Services offered	Two OS instances (Windows and Linux, RESTful APIs, Command Line Interface as well as Graphic User interface, data storage and networking	Offers multiple types of instances for a variety of purposes including General Purpose, Compute Optimised, Memory Optimised
Location	GCE servers are located in 4 main regions around the globe - Central USA, Western Europe, East Asia and Australia with automatic migration to another server if there is a hardware failure to the machine that is running the application	Amazon data centres are more available. They are located in 25 regions, including 3 Availability Zones in Sydney, with Melbourne coming soon.
General Purpose machines	Offers only 5 different tiers of General Purpose machines; E2, N2, N2D, N1, and Tau T2D	Offers up to 14 different tiered VM machines: Mac, T4g, T3. T3a, T2, M6g, M6i, M6a, M5, M5a, M5n, M5zn, M4, and A1

Compute Optimised machines	Can only select from: C2 machines	Again, offers a greater variety of machines: C7g, C6g, C6gn, C6i, C5, C5a, C5n, C4
Memory Optimised machines	Google currently offers 2 tiers of memory optimised machines: M2, and M1	Amazon has 15 Memory-optimised machines: R6g, R6i, R5, R5a, R5b, R5n, R4, X2gd, X2idn, X2iedn, X2iezn, X1e, X1, High Memory, and z1d

5. Examine Application Developer API and:
    a. Describe TWO processes for the input and output of data based on the company's API and how they achieve organisational objectives

Describe how each function (get and put requests) is achieved, how is a track saved, how you get the track

For requests and responses from Spotify's API, the following parameters are frequently encountered: Spotify ID - a base-62 identifier for an album, artist, playlist, track etc.

One of the processes when streaming audio files from Spotify is the ability of the current user to save tracks. A 'PUT request' is submitted, which will save one or more tracks in the user's "Your Music" library. When successful, a 200 response is given. However, for any unsuccessful attempts, an integer as well as a string will output e.g. "401, Bad or expired token. This can happen if the user revoked a token or the access token has expired. You should re-authenticate the user." This allows the user to efficiently store their saved songs in their own library and play tracks that they prefer to listen to when they use the Spotify service.

The user can use the 'GET' request to search their saved track. The 'search' will use the keyword string and match albums, artists, tracks, shows etc to Spotify's catalog information. This minimises the time for users to look for specific music that they wish to listen to.

Finally, the user is also able to remove tracks from their library. Similar to the PUT request that saves the track, in this case, the "DELETE" request can be used to remove the selected track. Upon successful deletion, the same response will be generated as the PUT request; "200, Track removed" or 400 for errors e.g. 401 or 403.

    b. Describe the functions of the API that facilitate your chosen feature or service

Summarise a description of e.g. save track list and get track list(need to have input and output)

From an organisational and functional perspective, Spotify has many functional endpoints possible based on its API. A feature of Spotify is its Search function. From the Web Player Client, a user can select "Search" on the GUI on the top left side. Prior to the user entering a string in the search bar, the app returns the following responses if there is an active connection between client and server through the API: "Recent Searches", "Your top

genres", and "Browse all". When entering a string into the search, Spotify's API initiates a GET request. The searches can be narrowed down using field filters, such as album, artist and track. A "NOT" operator can also be used to exclude keywords from the search. The maximum number of results returned to the user is set to a default unit of 20. E.g.

```
curl --request GET \
 --url 'https://api.spotify.com/v1/search?type=track,playlist&q=name:abcdef' \
 --header 'Authorization: ' \
 --header 'Content-Type: application/json'
```

The response of the GET request will display the following available objects; tracks, albums, artists, playlists, shows and episodes. Each object would contain the following key:value pair responses with its corresponding formats:

```
"Object": {
   "href": STRING,
   "items": [
     {ARRAY}
   ],
   "limit": INTEGER,
   "next": STRING,
   "offset": INTEGER,
   "previous": STRING,
   "total": INTEGER
 }
```

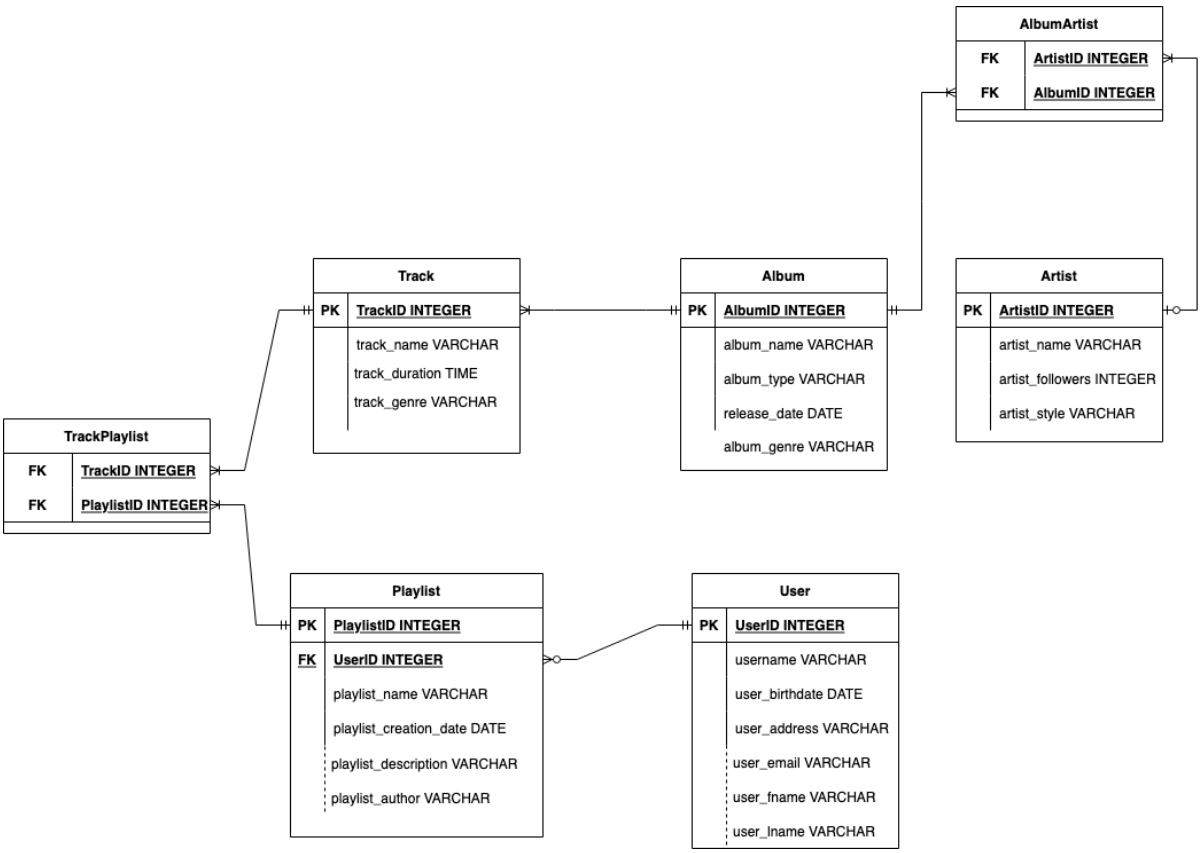
Another function of the Spotify API is to allow its users to change playlist details (assuming that they own the playlist). This can be achieved using the PUT/playlist/{playlist_id} request. Here, the "playlist_id" has its unique Spotify ID formatted as a string:

```
E.g. PUT/playlist/"3cEYpjA9oz9GiPac4AsH4n"

{
"name": STRING,
"public": BOOLEAN,
"description": STRING,
}
```

In the field of name, we would add the new name of the playlist e.g. "My new playlist". If "public" is *true*, the playlist will be made public and it will be private if *false*. Finally, any values entered in "description" will be displayed in Spotify Clients and in the Web API.

c. Create an entity relationship diagram to depict the database tables that you believe are likely to be employed by the system you are investigating in order to provide your chosen service or product. You are free to speculate on the properties of entities and the relationships between entities.



d. Supplementary to this diagram, suggest a way in which these entities might be altered in order to add an extra piece of functionality to the product or service. Explain how you think this would improve the service.

Similar to how youtube allows users to add chapters within their videos, we can add music sections to break down a track into different sections such as intro, outro, exposition, development, chorus, bridge, interlude or fadeout. Spotify could create a separate microservice in its API called ‘Spotify Studios’ where we can select any song from a selected Playlist to edit. Under the description of the track, we can add timestamps and titles (chorus, intro, outro, bridge etc). The time stamp should begin at 00:00, with each track having a minimum of 3 timestamps.

i. Explain what technologies or app functionality would need to be added to support this. Would this be difficult or expensive to implement?

ii. Create a modified entity relationship diagram that depicts your planned change

References

Hartman, J., 2021. Graphql vs rest: Difference between graphql and REST API. *Guru99*. Available at: https://www.guru99.com/graphql-vs-rest-apis.html [Accessed December 2, 2021].

Engineering, P.by S., 2021. Building the future of our desktop apps. *Spotify Engineering*. Available at: https://engineering.atspotify.com/2021/04/07/building-the-future-of-our-desktop-apps/ [Accessed December 2, 2021].

Meer, G.van der, 2021. How we use python at Spotify. *Spotify Engineering*. Available at: https://engineering.atspotify.com/2013/03/20/how-we-use-python-at-spotify/ [Accessed December 2, 2021].

Quora. 2021. *What is Spotify's architecture?*. [online] Available at: <https://www.quora.com/What-is-Spotifys-architecture> [Accessed 3 December 2021].

Amazon Web Services, 2021. *AWS re:Invent 2016: Amazon Global Network Overview with James Hamilton*. [video] Available at: <https://www.youtube.com/watch?v=uj7Ting6Ckk> [Accessed 3 December 2021].

Toea, A., 2021. *Beginning Frontend Development with React*. [online] Oreilly.com. Available at: <https://www.oreilly.com/live-events/beginning-frontend-development-with-react/0636920189800/0636920246619/> [Accessed 3 December 2021].

Amazon Web Services, Inc. 2021. *Data Centers - Our Data Centers*. [online] Available at: <https://aws.amazon.com/compliance/data-center/data-centers/> [Accessed 3 December 2021].

Zhai, L., 2021. *Tracking the Path of Spotify Music: Design Principles and Technologies that Make Spotify Workable | CCTP-820: Leading By Design*. [online] Blogs.commons.georgetown.edu. Available at: <https://blogs.commons.georgetown.edu/cctp-820-fall2019/2019/12/17/tracking-the-path-of-spotify-music-design-principles-and-technologies-that-make-spotify-workable/> [Accessed 3 December 2021].

Zenko. 2021. *Four critical differences between Google Cloud Storage and Amazon S3 APIs | Zenko*. [online] Available at: <https://www.zenko.io/blog/four-differences-google-amazon-s3-api/> [Accessed 3 December 2021].

Upguard.com. 2021. *Google Compute Engine vs Amazon's AWS EC2 | UpGuard*. [online] Available at: <https://www.upguard.com/blog/google-compute-engine-vs-amazons-aws-ec2> [Accessed 3 December 2021].

Amazon Web Services, Inc. 2021. *Global Infrastructure*. [online] Available at: <https://aws.amazon.com/about-aws/global-infrastructure/> [Accessed 3 December 2021].

Developer.spotify.com. 2021. *Web API Reference | Spotify for Developers*. [online] Available at: <https://developer.spotify.com/documentation/web-api/reference/#/operations/save-tracks-user> [Accessed 4 December 2021].

Konrad, A., 2021. *Why Spotify Really Decided To Move Its Core Infrastructure To Google Cloud*. [online] Forbes. Available at: <https://www.forbes.com/sites/alexkonrad/2016/02/29/why-spotify-really-chose-google-cloud/?sh=284571e33ee4> [Accessed 4 December 2021].

Docs.oracle.com. 2021. *What Are RESTful Web Services? - The Java EE 6 Tutorial*. [online] Available at: <https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html> [Accessed 4 December 2021].

Google Cloud. 2021. *Machine families | Compute Engine Documentation | Google Cloud*. [online] Available at: <https://cloud.google.com/compute/docs/machine-types> [Accessed 4 December 2021].

Google Cloud. 2021. *Storage classes | Google Cloud*. [online] Available at: <https://cloud.google.com/storage/docs/storage-classes#standard> [Accessed 4 December 2021].

Amazon Web Services, Inc. 2021. *Amazon EC2 Instance Types - Amazon Web Services*. [online] Available at: <https://aws.amazon.com/ec2/instance-types/> [Accessed 4 December 2021].

Jones, E., 2021. *Google Cloud vs AWS in 2021 (Comparing the Giants)*. [online] Kinsta®. Available at: <https://kinsta.com/blog/google-cloud-vs-aws/> [Accessed 4 December 2021].

Developer.spotify.com. 2021. *Web API Reference | Spotify for Developers*. [online] Available at: <https://developer.spotify.com/documentation/web-api/reference/#/operations/search> [Accessed 5 December 2021].

Developer.spotify.com. 2021. *Web API Reference | Spotify for Developers*. [online] Available at: <https://developer.spotify.com/documentation/web-api/reference/#/> [Accessed 5 December 2021].