 Version 0.1

Software Architecture Specification - Virtual Jukebox

BY

AUSTIN BEVACQUA, BRADLEY VAN DER ZWAN, DILLON VINCENT, RYAN WEBSTER, TANAKA CHITETE

# Change History

|  |  |
| --- | --- |
| Version | Notes |
| 0.1 | * First Iteration * Added each header from the provided template * Attempted majority of headers |

# Glossary

|  |  |
| --- | --- |
| Term | Definition |
| Framework | A platform for developing software applications. Contains in-built functionality to assist in the development of software. |
| Front-end | The front-end of an application is defined as what the users can see and directly interact with. |
| Back-end | The back-end of an application relates to the server-side tasks and database communication. |
| Database | A structured set of data held in a computer or server. |
| SDK  (Software Development Kit) | A collection of software development tools in one package. |
| API  (Application Programming Interface) | A software interface which offers a service to other pieces of software. |
| WebSocket | A web communication protocol allowing two-way communication between a client and a server. |
| FIFO | A process for ordering data, where the initial data that is added to a queue will be the data that is taken first from the queue. |

Contents

[**Change History**](#_bwu00cx13jnv)1

[**Glossary**](#_jpy9isf3la1)1

[**1 Introduction**](#_n9bdx1pa2en0)3

[1.1 Purpose](#_x4wecth77ukv) 4

[1.2 Audience](#_3o7alnk) 4

[1.3 Project Background](#_23ckvvd) 4

[**2 Business Requirements and Goals**](#_ttufs76f2gec)4

[2.1 Business Drivers](#_rd3hhi28f18g) 4

[2.2 Solution Requirements](#_mfsptkt837gq) 4

[2.3 High Level Metrics for Success](#_5atlij6qx1j) 5

[2.4 Architecture Boundary](#_8hgf0f883h6c) 5

[**3. Business Model**](#_ylfsy4n2v7uj)5

[3.1 Organisational Context](#_hfkoqxk1unb2) 29

[3.2 Current Situation](#_hfkoqxk1unb2) 29

[3.3 Solution Definition](#_hfkoqxk1unb2) 29

[3.4 Actors](#_hfkoqxk1unb2) 29

[3.5 Interactions of Actors](#_hfkoqxk1unb2)29

[3.6 Use Cases](#_vwat70acbry1) 9

[3.6.1 Logging In to a Virtual Jukebox Account](#_z8c3vkt9lynl) 9

[3.6.2 Hosting a Virtual Jukebox Session](#_8thxvmyd0r1w) 10

[3.6.3 Joining a Virtual Jukebox Session](#_5jeulzj1z8n5) 10

[3.6.4 Adding a Song to the Song Queue](#_976y1nbpb6n1) 11

[3.6.5 Voting for a Song in the Queue](#_d2ky5lb3xkjr) 11

[3.6.6 Using the Chat Room](#_ae0obribcwev) 12

[3.6.7 Closing a Virtual Jukebox Session](#_z5f0c0rbrbxg) 12

[**4 Solution Architecture Model**](#_hfkoqxk1unb2)29

[4.1 Solution Components](#_hfkoqxk1unb2)29

[4.1.1 Internal Components](#_qymyn79bwvuk) 13

[4.1.2 External Components](#_104sfzrxmmn8) 14

[4.2 Integration Specification](#_hfkoqxk1unb2)29

[4.3 Walk-Throughs](#_hfkoqxk1unb2)29

[4.3.1 Logging In to a Virtual Jukebox Account](#_bwghwjeqkl8g) 17

[4.3.2 Hosting a Virtual Jukebox Session](#_onvfida61nvn) 18

[4.3.3 Joining a Virtual Jukebox Session](#_9zm9cznw02d) 19

[4.3.4 Adding a Song to the Song Queue](#_fscvz9fdpn3z) 20

[4.3.5 Voting for a Song in the Queue](#_kw51nzupddmk) 20

[4.3.6 Using the Chat Room](#_b2p2zoau4wmw) 20

[4.3.7 Closing a Virtual Jukebox Session](#_f83i51k8a6k3) 21

[4.4 Information Model](#_a03bnm8mme5e) 21

[4.4.1 Virtual Jukebox Database](#_shq3rimracu) 21

[4.4.2 Redis cache](#_rkkvry4yr7pz) 22

[**5 Detailed Physical Architecture**](#_hfkoqxk1unb2)29

[5.1 Network Diagram](#_hfkoqxk1unb2) 29

[5.2 System Specification Summary](#_hfkoqxk1unb2) 29

[5.3 Configuration Requirements](#_hfkoqxk1unb2)29

[**6 Solution Impact**](#_tavmc54h2eym)24

[6.1 Impact on Processes](#_hfkoqxk1unb2)29

[6.2 Impact on People](#_8dn7su5f0h0d) 24

[6.3 Technical Impact](#_em6d5vp3hbgo) 24

[**7 Quality Attributes**](#_sjqv3lgleli)25

[7.1 Reliability](#_b2zlxy3mzhda) 25

[7.2 Availability](#_hfkoqxk1unb2) 29

[7.3 Serviceability](#_hfkoqxk1unb2) 29

[7.4 Best Practice Commentary](#_hfkoqxk1unb2) 29

[7.5 Architecture Commentary](#_hfkoqxk1unb2)29

[**8 Success Criteria and Test Plan**](#_hfkoqxk1unb2)29

[8.1 Use Case Validation](#_hfkoqxk1unb2) 29

[8.2 Validation of Solution metrics](#_hfkoqxk1unb2) 29

[8.3 Test Plan](#_hfkoqxk1unb2)29

[**9. Phasing, Roadmap, and Implementation Plan**](#_hfkoqxk1unb2)29

[9.1 Technical Assumptions](#_hfkoqxk1unb2) 29

[9.2 Solution Phases and Technical Milestones](#_hfkoqxk1unb2) 29

[9.3 Technical Work BreakDown Structure](#_hfkoqxk1unb2) 29

[9.4 Implementation Instructions](#_hfkoqxk1unb2)29

[**10 Appendix**](#_hfkoqxk1unb2)29

[**11 References**](#_17tt6393l9k0)29

# 

# 

## 1 Introduction 1.1 Purpose

This report …

## 1.2 Audience

This report is aimed at the stakeholders involved in the development of the Virtual Jukebox application. The main stakeholder is Amristar, which is the supervisor and client of the project. Stakeholders also include the staff involved with Computing Capstone Project 1 at Curtin University, which act as the co-supervisors, and the Developers of the application.

## 1.3 Project Background

Jukeboxes were once a common sight at bars, discos and any social gathering. They allowed users to choose a song from a playlist to add to a queue that would eventually be played at the gathering. However, due to the explosive rise of music streaming services, jukeboxes have since become nothing more than a novelty. The Virtual Jukebox application aims to reproduce the functionality of a physical jukebox in a digital, web application format. Users will be able to connect to a local jukebox by selecting a location on a map or by scanning a QR code, and be able to vote for songs from a playlist which will be added to the queue.

# 2 Business Requirements and Goals

## 2.1 Business Drivers

A gap in the market has been identified, with guests in attendance of venues currently having very little control over the music that is being played. Businesses are commonly playing music from predefined playlists in a random order, or broadcasting a radio station with no control over what is being played. If users are unhappy with the music, the only option is to find the person responsible for the music and to request that the song be skipped, or a different song be played. The Virtual Jukebox application aims to solve this issue by providing guests with the ability to control the music that is being played at a venue.

## 2.2 Solution Requirements

The Virtual Jukebox application will allow hosts to create a Virtual Jukebox account and login using their credentials. The application will also allow guests to generate and use a guest account to join a session, which will be generated with a random username. Guests will be able to connect to a Virtual Jukebox Session either by scanning a session’s QR code (or directly visiting the link), or by selecting a session on the Virtual Jukebox map. Each Virtual Jukebox session will have a playlist acting as a pool of songs which users can choose from to add to the song queue, which will be imported from the host’s Spotify account.

The Virtual Jukebox application will utilise a song queue which is viewable by all guests as well as the host. The queue will control the ordering of song playback on the host device. The user-queued songs will be ordered in a first-in-first-out (FIFO) manner by default, but songs with votes will be prioritised in playback order. Songs with the same number of votes will remain in a FIFO order. When no songs are queued by users, the host device will randomly play songs in the host defined playlist. Every session will allow all users to exchange instant messages through the live chat-room, with these messages being filtered for profanity.

## 2.3 High Level Metrics for Success

\*\* Unsure if this refers to measurements once the application has been deployed such as: number of venues/hosts using the application, percentage of guests in a venue connected to the session, number of votes cast, number of messages sent. Or if it refers to a certain percentage of the functional and non-functional requirements implemented (e.g., if all ‘high’ business value requirements are implemented, 40/49 functional requirements successfully implemented)

## 2.4 Architecture Boundary

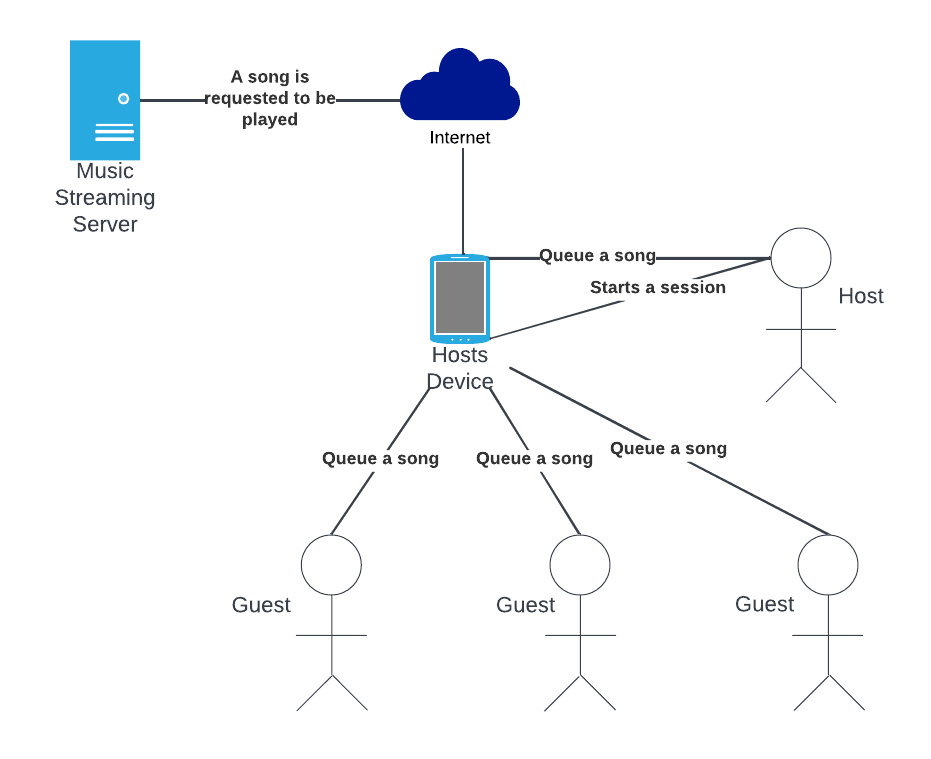
“What is in-scope, and what is not in-scope.”

# 3. Business Model

## 3.1 Organisational Context

The current state of music streaming exists primarily for an individual. Music can be instantly streamed for an individual through various music streaming services such as Spotify, which can instantly playback any song out of a database of millions. When it comes to music streaming for a group of people, options are extremely limited. The current situation of group music streaming for gatherings, social events, or businesses is to have a central ‘host device’ in which guest users can interact directly with to queue songs.

\*\*Fully aware this diagram is not detailed enough. What more could be added?



## 3.2 Current Situation

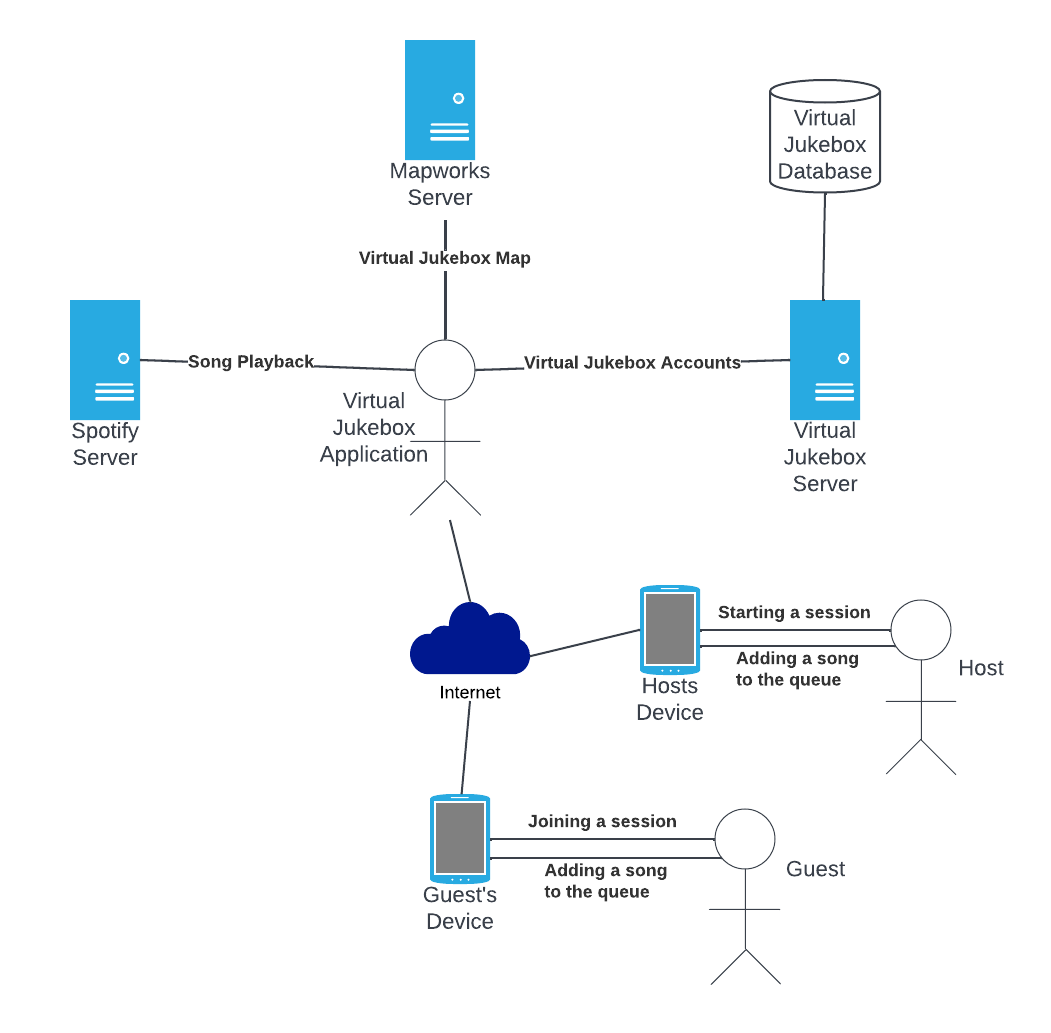
\*\*TODO in draft 2

* Essentially describe the same thing, but add a “story”
* More specific user interactions
* More detail. Personify different components. Paint a scenario (e.g. house party)

## 3.3 Solution Definition

\*\*Unsure as to what level of detail this should be

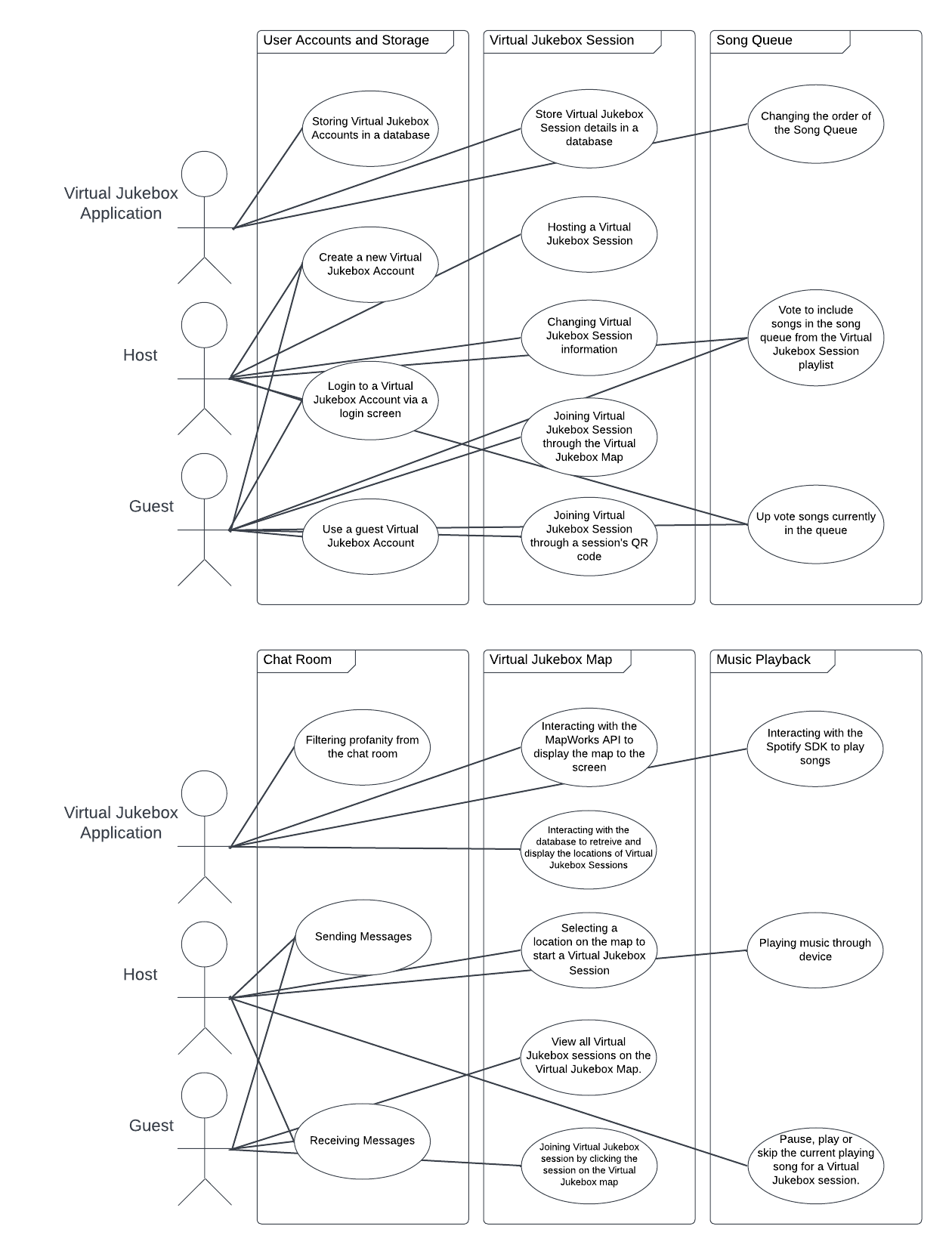
\*\*Description of diagram to be added in draft 2



## 3.4 Actors

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | **Description** | **Current use of the System** | **Expected use of the New Solution** |
| **Virtual Jukebox Application** | The application which acts as a Jukebox. | Does not exist | The Virtual Jukebox application is responsible for providing users the ability to set up and join Virtual Jukebox sessions, managing these sessions’ song queues, and in turn, handling music playback via Spotify |
| **Hosts** | A type of user responsible for | Hosts will control playback through their device, and will physically pass their device to guests users for the guest users to queue songs. Hosts can additionally queue songs on their own device. | Hosts may create a Virtual Jukebox session, and can control the music playback. Hosts can vote for songs in the song queue, and add songs from the session playlist into the song queue. |
| **Guests** | A type of user who does not have control over the song queue | Guests will not have any direct control over the song queue unless they are able to interact with a host user’s device, when they can then queue songs on. | Guests can create and log into an account, join a session, queue songs in a session, send messages, and vote on the current song queue. Guests are not required to log into an account to join sessions. |

## 3.5 Interactions of Actors

\*\*Is there a better way to lay this out?

\*\*Should there be more interactions per product function?

## 3.6 Use Cases

\*\*Use case diagrams will be added in draft 2

\*\*Wording for user stories is certainly not final.

### 3.6.1 Logging In to a Virtual Jukebox Account

As a HOST I want to LOG INTO MY VIRTUAL JUKEBOX ACCOUNT so I can HOST A VIRTUAL JUKEBOX SESSION

**Goal**: To allow a host to login to their Virtual Jukebox Account

**Primary Actor**: Hosts

**Secondary Actor**: Virtual Jukebox Application

**Precondition**: The host is connected to the internet

**Trigger**: The host selects the “login” button

**Flow of Events**:

1. The host enters their username and password of their Virtual Jukebox Account
2. The host clicks the submit button
3. The system checks that the hashed password is identical to the one stored in the database
4. The system notifies the user that they have successfully logged in
5. The system closes the login screen and opens the home page.

**Extensions:**

* 1A - The host has not created a Virtual Jukebox Account

1. The user will select the make an account button
2. The user will enter a username, email and password to assign to the new Virtual Jukebox Account
3. The user will connect their Virtual Jukebox Account to an existing Spotify Premium account
4. The use case resumes at step 4

* 1B - The host does not remember the password for an existing Virtual Jukebox Account

1. The user will select the reset password button
2. The user will enter the email of their Virtual Jukebox Account
3. The system will send an email containing a password reset link to the provided email.
4. The use case resumes at step 1

* 3A - The system cannot find the account in the database

1. The system notifies the user that there is an incorrect username or password
2. The use case resumes at step 1

* 3B - The system finds the hashed password provided and the one stored are different

1. The system notifies the user that there is an incorrect username or password
2. The use case resumes at step 1

### 3.6.2 Hosting a Virtual Jukebox Session

As a HOST I want to CREATE A VIRTUAL JUKEBOX SESSION so I can ALLOW OTHER USERS TO JOIN THE SESSION

**Goal**: To allow hosts to create a Virtual Jukebox Session

**Primary Actor**: Hosts

**Secondary Actor**: Virtual Jukebox Application

**Precondition**: The actor is connected to the internet and logged into their account

**Trigger**: The host selects the start a session button

**Flow of Events**:

1. The system retrieves the current location of the host device
2. The system prompts the user to select a location on the Virtual Jukebox Map to host the Virtual Jukebox Session. The initial location of the map is the location retrieved at step 1.
3. The system prompts the user to enter a name for their Virtual Jukebox Session
4. The system prompts the user to enter a description for their Virtual Jukebox Session
5. The system interacts with the Spotify SDK to fetch all of the host’s Spotify Playlists
6. The system prompts the user to select a playlist for their Virtual Jukebox Session
7. The system adds the Virtual Jukebox Session information to the database

**Extensions:**

* 1A - The host’s device does not have location services enabled

1. The use case resumes at step 2

* 6A - The host’s connected Spotify account does not contain any playlists

1. The system notifies the user that they have no active playlists and asks the user to create a playlist inside of Spotify
2. The use case ends

### 3.6.3 Joining a Virtual Jukebox Session

As a GUEST I want to JOIN A VIRTUAL JUKEBOX SESSION so I can VOTE ON THE SONGS PLAYING AND ACCESS THE CHAT SCREEN

**Goal**: To allow guests to join a Virtual Jukebox Session

**Primary Actor**: Guests

**Secondary Actor**: Virtual Jukebox Application

**Precondition**: The guest is connected to the internet

**Trigger**: The guest selects a Virtual Jukebox Session from the Virtual Jukebox Map or scans a room’s QR code

**Flow of Events**:

1. The system will display the title and description of the Virtual Jukebox Session to the guest
2. The guest will click the join button to join the Virtual Jukebox Session
3. The system will connect the user to the Virtual Jukebox Session

**Extensions:**

* 2A) TheVirtual Jukebox Session is passworded

1. The guest will be prompted to enter the password for the Virtual Jukebox Session
2. The system checks that the hashed password is identical to the one stored in the database
3. If the password is correct, the use case resumes at step 3.
4. If the password is incorrect, the use case resumes at step 2A.1

### 3.6.4 Adding a Song to the Song Queue

As a GUEST OR HOST I want to ADD A SONG FROM THE VIRTUAL JUKEBOX SESSION PLAYLIST TO THE SONG QUEUE so I can HAVE MY FAVOURITE SONG PLAY

**Goal**: To allow hosts and guests to add songs from a Virtual Jukebox Sessions’s playlist to the session’s song queue.

**Primary Actor**: Guests and Hosts

**Secondary Actor**: Virtual Jukebox Application

**Precondition**: The guest or host is connected to the internet, and is currently connected to an active Virtual Jukebox Session

**Trigger:** A guest or host clicks the view playlist button

**Flow of Events**:

1. The system shows the guest or host all songs currently in the Virtual Jukebox Session’s playlist
2. The user selects a song from the playlist to add to the queue
3. The system checks the number of credits the user has in the database.
4. The system decreases the number of credits the user has by one
5. The system adds the selected song to the end of the Virtual Jukebox Session’s song queue

**Extensions:**

* 2A) The selected song is already in the song queue

1. The system notifies the user that the current song is already in the queue
2. The use case resumes at step 1

* 3A) The guest or user does not have any credits

1. The system notified the user they do not currently have any credits to add songs to the song queue
2. The use case resumes at step 1

### 3.6.5 Voting for a Song in the Queue

As a GUEST OR HOST I want to VOTE ON A SONG IN THE SONG QUEUE so I can LISTEN TO MY FAVOURITE SONGS FASTER

**Goal**: To allow hosts and guests to upvote songs currently in a Virtual Jukebox Sessions’s song queue.

**Primary Actor**: Guests and Hosts

**Secondary Actor**: Virtual Jukebox Application

**Precondition**: The guest or host is connected to the internet, and is currently connected to an active Virtual Jukebox Session

**Trigger:** A guest or host clicks the view queue button

**Flow of Events**:

1. The system shows the guest or host all songs currently in the Virtual Jukebox Session’s song queue
2. The user selects a song from the list to upvote
3. The system will add the vote to the database
4. The system will reorder the current song queue by the total number of votes on each song
5. The system will push the updated song queue to all connected users.

**Extensions:**

* 1A) There are no songs currently in the song queue

1. The system will notify the user there are currently no songs in the song queue
2. The use case ends

* 2A) The user has already upvoted the selected song

1. The use case resumes at step 1

### 3.6.6 Using the Chat Room

As a GUEST OR HOST I want to SEND MESSAGES IN THE CHAT ROOM so I can MESSAGE ALL USERS IN THE VIRTUAL JUKEBOX SESSION

**Goal**: To allow hosts and guests to send messages within a Virtual Jukebox Session.

**Primary Actor**: Guests and Hosts

**Secondary Actor**: Virtual Jukebox Application

**Precondition**: The guest or host is connected to the internet, and is currently connected to an active Virtual Jukebox Session

**Trigger:** A guest or host clicks the view chat button

**Flow of Events**:

1. The system shows the user all messages they have received since joining the Virtual Jukebox Session
2. The user selects the blank message box to start typing a new message
3. The user finishes typing their message and clicks the send message button
4. The system checks the message typed by the user for profanity
5. The system sends the message to all users currently connected to the Virtual Jukebox Session

Extensions:

* 4A) The system detects profanity in the typed message

1. The system will alert the user not to use profane language
2. The system will replace the detected profanity with asterisk symbols
3. The use case resumes at step 5

### 3.6.7 Closing a Virtual Jukebox Session

As a HOST I want to CLOSE A VIRTUAL JUKEBOX SESSION so

**Goal**: To allow hosts to close a Virtual Jukebox Session

**Primary Actor**: Hosts

**Secondary Actor**: Virtual Jukebox Application and Guests

**Precondition**: The host is connected to the internet, and is currently connected to an active Virtual Jukebox Session

**Trigger:** The host selects the session settings button

**Flow of Events**:

1. The user selects the “end session” button
2. The system asks the user if they are sure they want to close the Virtual Jukebox Session
3. The user selects “yes”
4. The system disconnects all guests from the active Virtual Jukebox Session
5. The system removes the session from the database
6. The system notifies the host the session has been closed

# 4 Solution Architecture Model

## 4.1 Solution Components

### 4.1.1 Internal Components

**Virtual Jukebox session:** A Virtual Jukebox session is an instance of a song queue with an associated playlist and host, which guest users can join. Each session has one song playing at any given time (unless stopped or paused by the host), and the song queue will dictate which song will play next. Virtual Jukebox Sessions may be public, meaning they are joinable by any guest user, or may be private, meaning they are only joinable by guest users with access to a password or QR code. All Virtual Jukebox Sessions have an associated connection range in metres, in which guest users are unable to join public or private sessions without being within that session’s connection range.

**Song queue:** A song queue is a data structure abstracted to all users as an ordered list of songs. The song queue acts as a priority queue, where a higher priority moves elements to the front of the queue. Each element in the song queue is a song, and ordering shifts in accordance to user and Virtual Jukebox session actions such as queuing songs (appends a song), up-voting a song (bumps the song up the queue), and playing a song (removes the front song from the queue). Given no user actions other than queuing songs, the song queue operates in a first-come-first-serve (FIFO) manner. A song queue cannot contain two of the same song at any given time. One song queue is linked to a Virtual Jukebox session, but song queues come in two separate types as follows:

* **User-declared queue**: A user-declared queue exists when any user in a Virtual Jukebox session queues a song. This song queue allows all users within the Virtual Jukebox session to up-vote songs, whereas the ambient queue does not.
* **Ambient queue**: An ambient queue exists when and only when there does not exist a user-declared queue. In other words, the ambient queue plays when no songs are queued by users. This queue consists of the entire playlist of the Virtual Jukebox session shuffled.

**Song:** A song is a Spotify-based track, identified by its Spotify ID. Songs can be in a playing state (i.e. currently streaming) or not (queued or part of a playlist) at any given time. Every song stored in the Virtual Jukebox application is part of a playlist, but can be present in song queues and the currently playing song within a Virtual Jukebox session.

**Playlist:** A playlist is a collection of songs. One playlist is associated with a Virtual Jukebox session at any given time which forms the pool of songs that all users are able to choose from to queue. Hosts are able to import Spotify playlists linked to their Spotify Premium account to be used as a playlist for a Virtual Jukebox session.

**Chat room:** The chat room is the Virtual Jukebox application’s component referring to its messaging system. A separate chat room instance exists for each Virtual Jukebox session, which enables all users connected to a session to view chat history (of the session) from the point at which they have joined the session onwards. All users can post textual messages in a chat room, and all users may also view those messages. The chat room includes the following moderation functionality:

* **Message rate limiting**
* **Profane language filtration**

**Virtual Jukebox** **database (PostgreSQL):** The primary database for persistent storage of the Virtual Jukebox application. The database is PostgreSQL-based and situated on the Virtual Jukebox web server. The database is responsible for securely storing Virtual Jukebox account data including user passwords, usernames and emails, as well as Virtual Jukebox sessions data including geographical locations, and connection ranges.

**Virtual Jukebox web server:** The Virtual Jukebox web server is a web server responsible for storing the PostgreSQL database. This web server resides on the Amristar server (in the network diagram section 5.1).

**Redis cache:** A server-side cache storing commonly accessed data items in the Virtual Jukebox application.

**Web page view:** The web page view refers to the visible front-end portion of the Virtual Jukebox application. As the application is a single-page web application, the web page view will be responsible for displaying all Virtual Jukebox related information to all users. This includes the:

* **Virtual Jukebox session screen**, which includes the following views:
  + Settings page view
  + Chat room view
  + Song queue view
  + Playlist selection page view
* **Virtual Jukebox session join screen**
* **Virtual Jukebox dashboard screen**, which has options to find a Virtual Jukebox session or create one (if logged in).
* **Virtual Jukebox map selection screen,** which has different views depending on guest or host use case:
  + Guest users use this screen to select a Virtual Jukebox session for joining
  + Host users use this screen to select a location for the Virtual Jukebox session.
* **Virtual Jukebox login and registration screens**

### 4.1.2 External Components

**Music streaming system:** Also referred to as the Spotify SDK. The music streaming system refers to Spotify’s playback system which is utilised by the Virtual Jukebox application to enable host music playback. The Spotify SDK is responsible for enabling this user playback via API calls to Spotify’s web server.

**Spotify web server:** The web server used by Spotify to store user account information and track information. The Virtual Jukebox application isn’t responsible for this web server, and assumes it exists within Spotify.

**Spotify database:** The database residing on the Spotify web server storing Spotify account information and track information. The Virtual Jukebox application isn’t responsible for this database, and assumes it exists within Spotify.

**Virtual Jukebox map system:** Also referred to as the Mapworks SDK. The Virtual Jukebox map system refers to the mapping software development kit i.e. the Mapworks SDK, which is used for sourcing geographical data used for displaying Virtual Jukebox sessions, via API calls to the Mapworks web server.

**Mapworks web server:** The web server used by Mapworks to store geographical information and provide mapping services. The Virtual Jukebox application isn’t responsible for this web server, and assumes it exists within Mapworks.

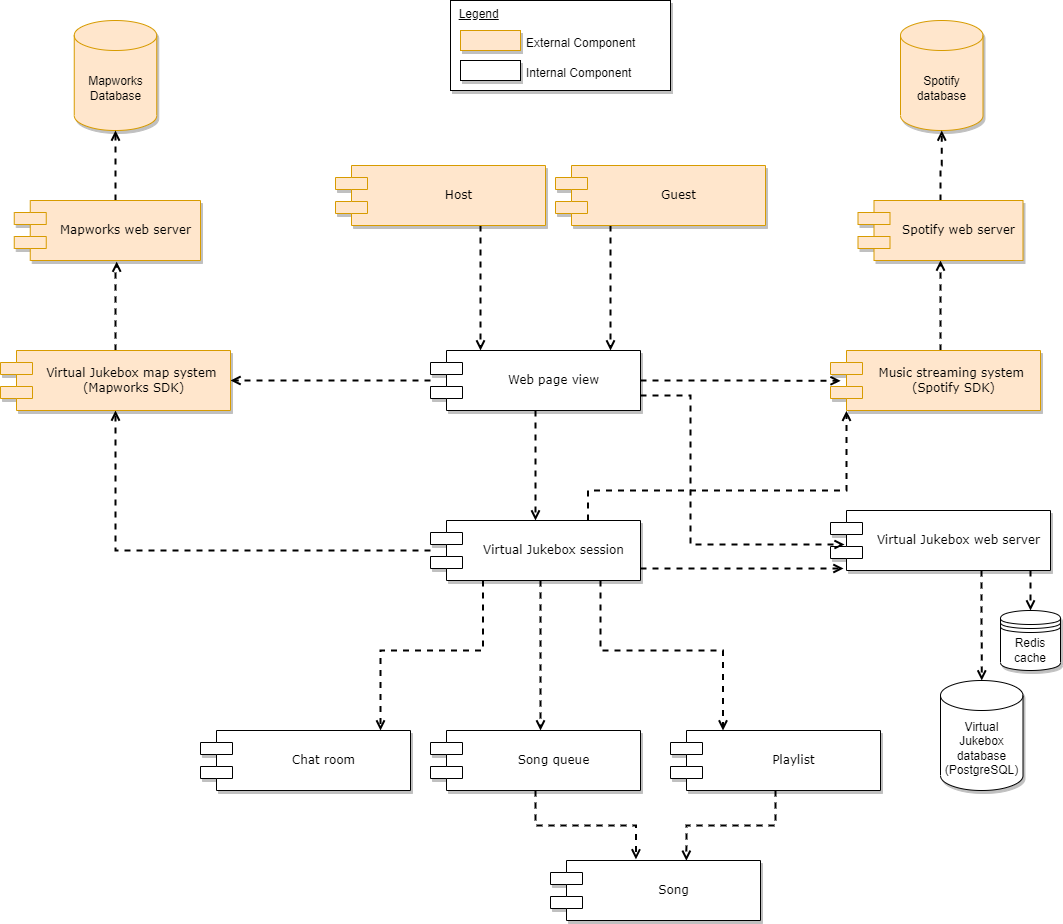
**Mapworks database:** The database residing on the Mapworks web server storing geographical information and mapping services. The Virtual Jukebox application isn’t responsible for this database, and assumes it exists within Mapworks.

**Host:** (See Actors sections for a detailed description). A type of user hosting a Virtual Jukebox session. Hosts require a Virtual Jukebox account as well as a Spotify Premium account to enable Spotify SDK usage. Also has all the functionality of guests while in a Virtual Jukebox session.

**Guest:** (See Actors sections for a detailed description). A type of user that can queue songs in a Virtual Jukebox session using credits, and upvote songs in the song queue.

## 4.2 Integration Specification

The following diagram showcases how each component interacts with and depends on each other.



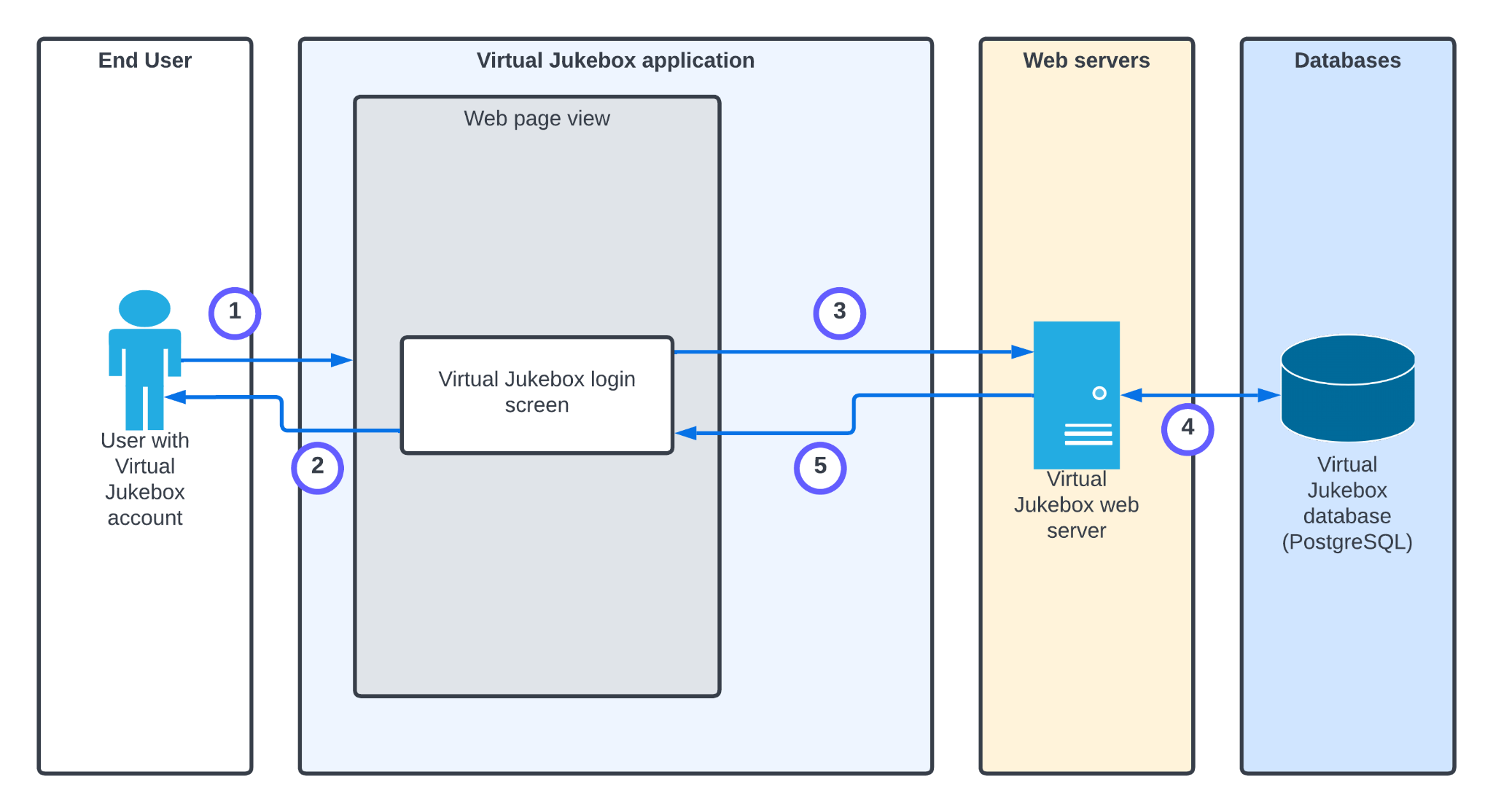
Host and Guest functionality depends on the web page view, as host and guest users are unable to use the application without a front-end web page. Hosts use the web page view to log into their accounts and create Virtual Jukebox sessions, whereas guests use the web page view to search for and join Virtual Jukebox sessions. The web page view depends on Virtual Jukebox sessions as it serves as the interface for collecting and showcasing Virtual Jukebox sessions data and components (such as the chat room and song queue). The web page view also depends on (1) the Spotify SDK to enable host Spotify account verification, (2) the Virtual Jukebox map system to show a map of Virtual Jukebox sessions to users, and (3) the Virtual Jukebox web server to fetch the Virtual Jukebox session data itself in addition to verifying host account login credentials. The Virtual Jukebox map system and the music streaming system depend on their respective web servers and in turn, their respective databases to provide the Virtual Jukebox application with mapping functionally and Spotify integration respectively. The Virtual Jukebox web server is dependent on the database and the Redis cache to act as a data source for Virtual Jukebox application functions such as user login validation.

Virtual Jukebox sessions are dependent on (1) the Virtual Jukebox map system for enforcing session connection ranges, (2) a chat room to provide messaging functionality for connected users, (3) the song queue to provide functionality in song orders, (4) playlists to enable guests to request songs to queue out of a pool of songs specified by the host, (5) The Virtual Jukebox web server to store and retrieve session data (such as session passwords and session connection ranges), and (6) the music streaming system to enable music playback of the current playing song. Both the song queue and playlist depend on songs to enable their respective functionality (as both act as song containers).

## 4.3 Walk-Throughs

### 4.3.1 Logging In to a Virtual Jukebox Account

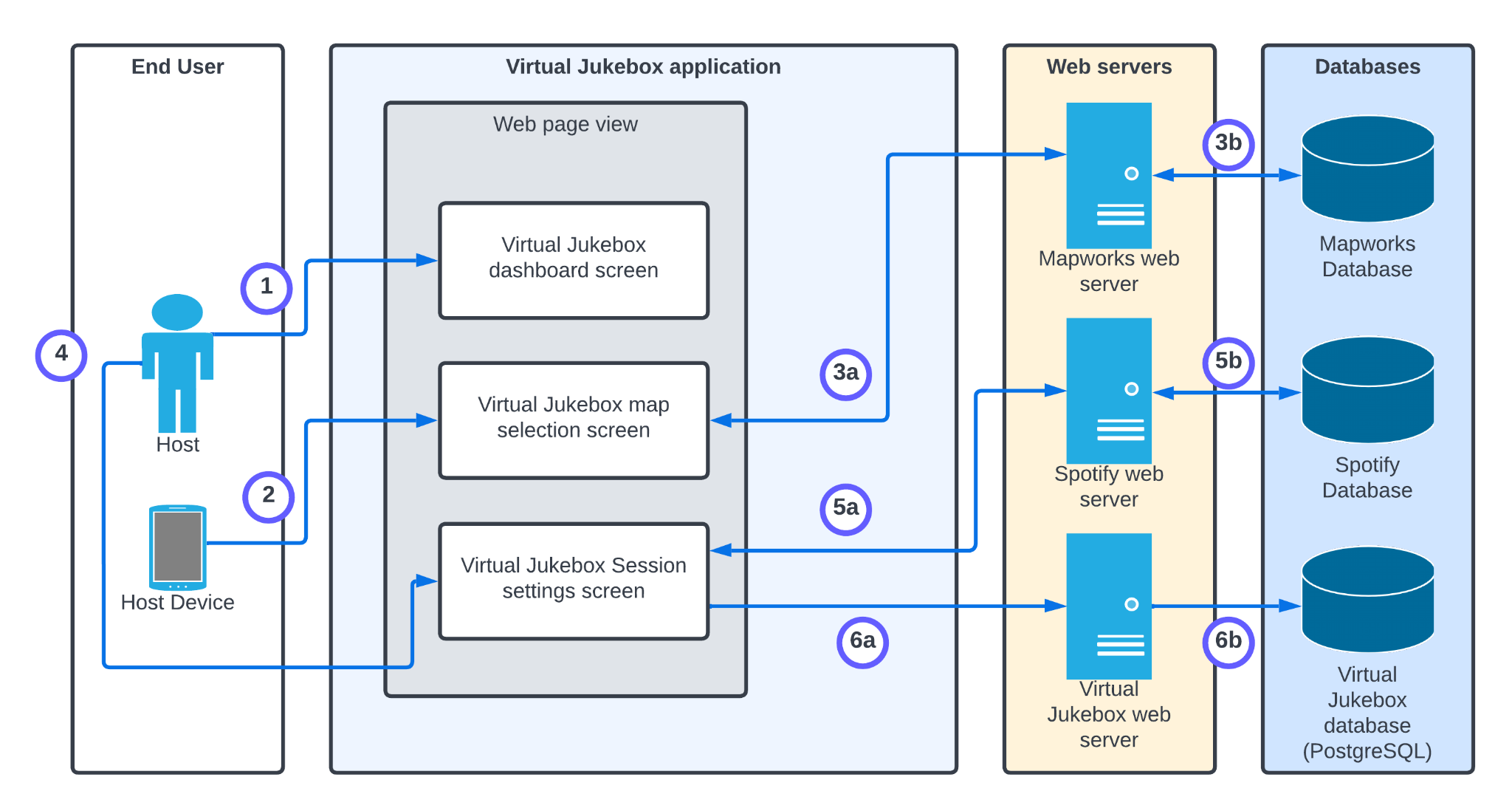
This activity involves a user of the Virtual Jukebox application logging into their account given they have previously registered an account.



|  |  |
| --- | --- |
| **Step** | **Description** |
| 1. | The user accesses the Virtual Jukebox application through a mobile or desktop browser. |
| 2. | The web application is launched and displays login fields including username and password fields for the user. |
| 3. | After providing login credentials, a request is sent to the Virtual Jukebox web server for credential validation. |
| 4. | A query is made to the Virtual Jukebox database to check whether the usernames provided matches any of the usernames stored. If so, the provided password is checked with the corresponding password linked to the stored email matched. |
| 5. | On success, the Virtual Jukebox dashboard screen is displayed on the web page. If the password did not match, or the username was not registered, an error message is displayed to the web page. |

### 4.3.2 Hosting a Virtual Jukebox Session

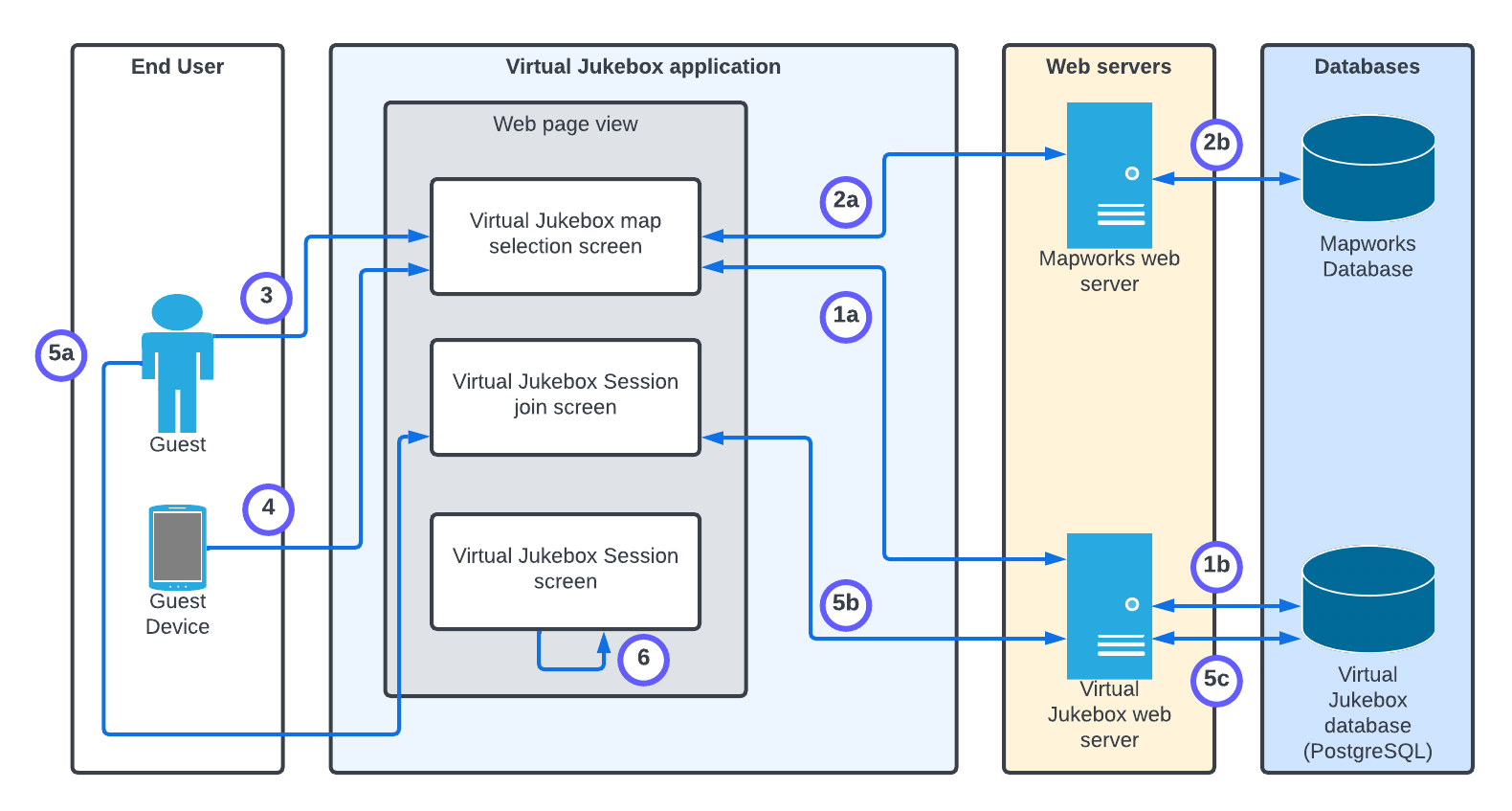
This activity involves a user of the Virtual Jukebox application with an account in setting up a Virtual Jukebox session to become a host.



|  |  |
| --- | --- |
| **Step** | **Description** |
| 1. | The logged in user selects the start a session button |
| 2. | The system retrieves the current location of the host device via GPS if available. |
| 3a + 3b. | The system makes API calls via the Mapworks SDK to display the Virtual Jukebox map with the current location selected in which users can select a different location if wanted or needed (e.g. no GPS).  (3a - Mapworks web server access via API calls, 3b - accessing database (done by Mapworks web server)) |
| 4. | The system prompts the user to enter the Virtual Jukebox session’s name and description. |
| 5a +  5b. | The system displays the user’s Spotify playlists by making API via the Spotify SDK, to be selected by the user.  (5a - Spotify web server access via API calls, 5b - Spotify database playlist retrieval (done by Spotify web server)) |
| 6a + 6b. | The system adds a record to the Virtual Jukebox database, storing the newly created Virtual Jukebox session. For the duration of the Virtual Jukebox being active, the playlist data (including songs) will be stored on the Redis cache.  (6a - Virtual Jukebox web server access, 6b - writing to the database and Redis cache) |

### 4.3.3 Joining a Virtual Jukebox Session

This activity involves a guest of the Virtual Jukebox application joining a Virtual Jukebox session via the Virtual Jukebox map or by scanning a QR code. Note that it is assumed the guest user is using a device with a GPS and a camera, and that the user has selected the join a session button.



|  |  |
| --- | --- |
| **Step** | **Description** |
| 1a + 1b. | The system queries the Virtual Jukebox database via the Virtual Jukebox web server (using Django ORM) to retrieve all active Virtual Jukebox data.  (1a - Virtual Jukebox web server access , 1b - reading from the database) |
| 2a + 2b. | The Virtual Jukebox map system is used to display all active Virtual Jukebox sessions on a map for the user.  (2a - Mapworks API calls to the Mapworks web server, 2b - accessing database (done by Mapworks web server)) |
| 3. | The user either selects the Virtual Jukebox session on the map selection screen (displayed using the Mapworks API), or scans a QR code (using their device’s camera). |
| 4. | On Virtual Jukebox session selection, the system uses the guest device’s GPS to check whether the selected session is within connection range of the device (queries database of stored session geographical location data). |
| 5a + 5b + 5c. | The guest enters a password if necessary. This password is hashed and checked with the hash of the corresponding session’s password stored on the Virtual Jukebox database.  (5a - guest provides the password, 5b/5c - Virtual Jukebox web server and database access with password authentication). |
| 6. | The Virtual Jukebox session’s data is loaded on the Virtual Jukebox session screen. |

### 4.3.4 Adding a Song to the Song Queue

This activity involves a guest or host user queuing a song within a Virtual Jukebox session. It assumes guests have already joined the session.

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1. | The system displays a list of all songs in the playlist of the Virtual Jukebox session (playlist selection view) by accessing the playlist via the Redis cache. |
| 2. | The user selects a song in the playlist. |
| 3. | The system queries the amount of credits the user has on their account for the Virtual Jukebox session by querying the Redis cache. |
| 4. | If the amount of credits assigned to the user is > 0, the system decrements that credit counter. |
| 5. | The system writes to the song queue via a Redis cache operation to add the song to the end of the queue. |
| 6. | The system pushes the updated song queue to all user’s web page views in the Virtual Jukebox session. |

### 4.3.5 Voting for a Song in the Queue

This activity involves a guest or host user voting for a song within the song queue given it is a user-declared song queue, in order to increase the priority of that song in the queue.

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1. | A user selects the up-vote button next to a song in the song queue. |
| 2. | An update Redis operation is undergone to increase the priority of the selected song in the song queue. . |
| 3. | The system pushes the updated song queue to all user’s web page views in the Virtual Jukebox session. |

### 4.3.6 Using the Chat Room

This activity involves a guest or host user posting and reading messages in the chat room while in a Virtual Jukebox session.

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1. | TODO |
|  |  |
|  |  |

### 4.3.7 Closing a Virtual Jukebox Session

This activity involves a host closing an active Virtual Jukebox session, disabling guests from seeing or accessing the session which will also remove all contents from the song queue.

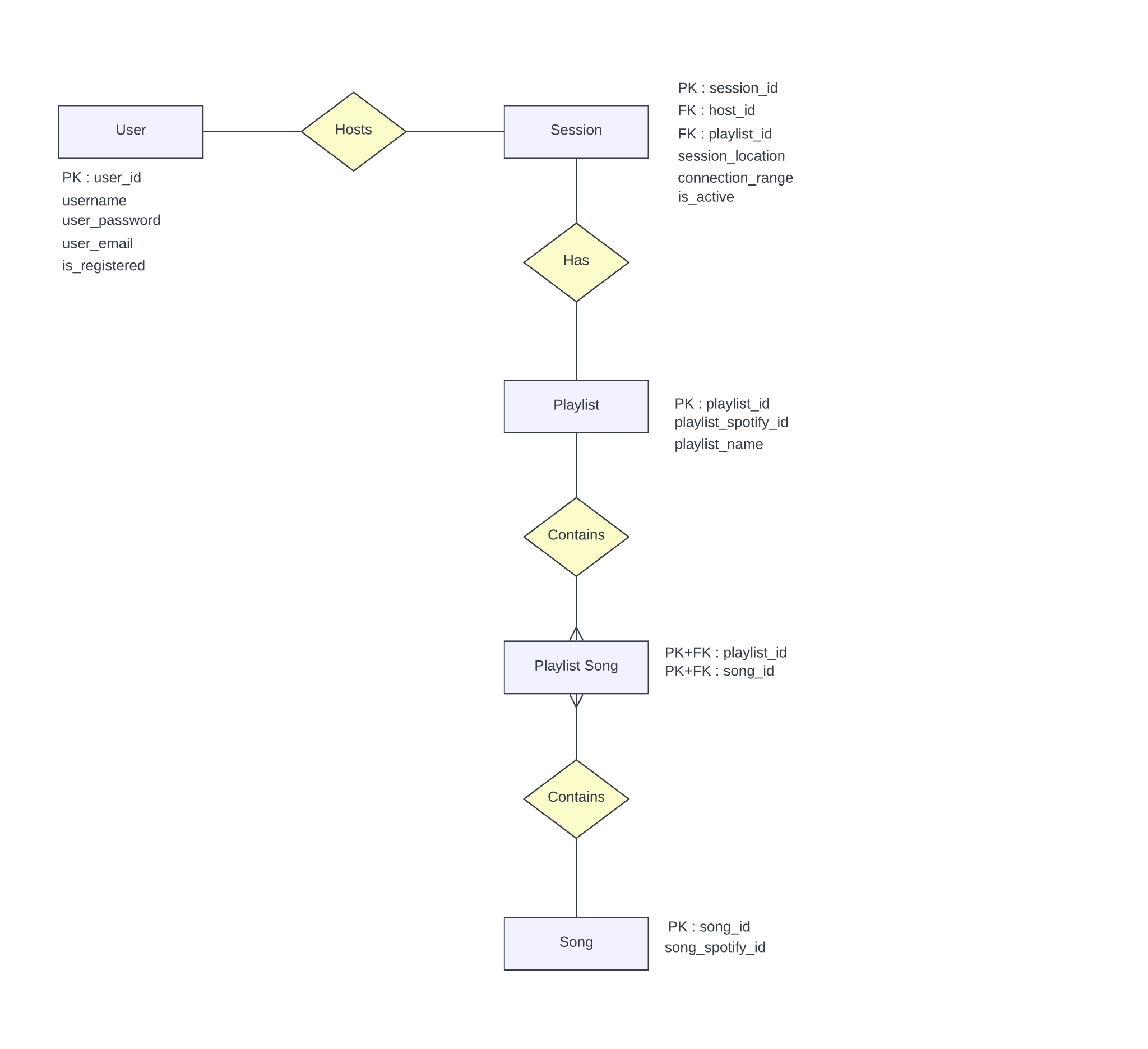
|  |  |
| --- | --- |
| **Step** | **Description** |
| 1. | TODO |
|  |  |
|  |  |

## 4.4 Information Model

### 4.4.1 Virtual Jukebox Database

The following Entity Relationship Diagram (ERD) outlines the information model of the Virtual Jukebox database, including its tables, attributes and relationships.

Note that “Virtual Jukebox” has been omitted from these table names that represent components that would usually include it. For example, “Virtual Jukebox sessions” are represented by the “Session” table.

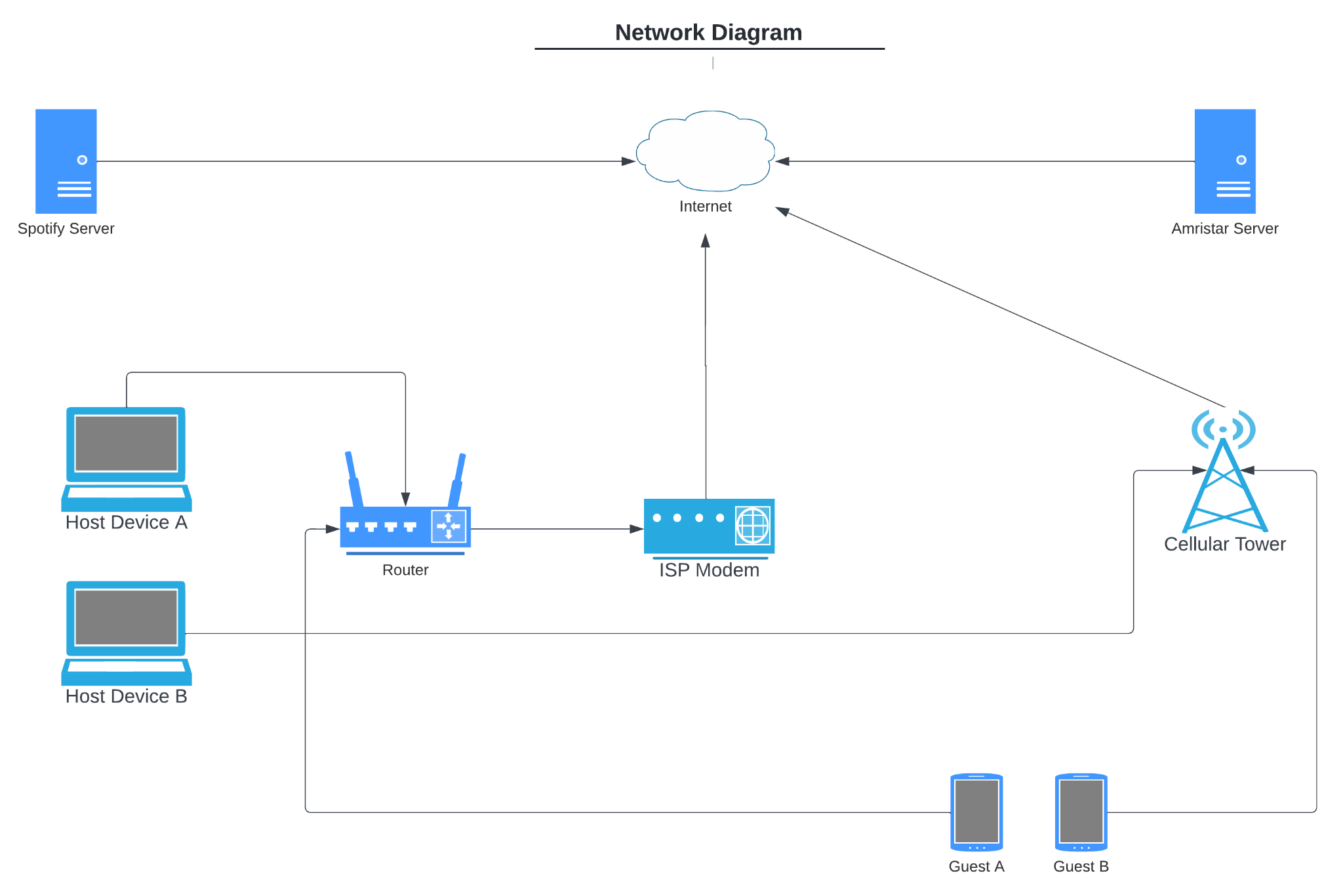


### 4.4.2 Redis cache

* Each song queue is stored on the Redis cache/on-memory, as there is no need for persistence.
* Credits for each user will reset when they leave a Virtual Jukebox session, therefore there is no reason to make them persistent, meaning their counters will reside on the Redis cache/on-memory.
* The playlist of a Virtual Jukebox session is stored on the Redis cache in addition to the database, due to its frequently accessed nature, and the fact that it wouldn’t be changed often (synchronisation not a concern).

# 5 Detailed Physical Architecture

## 5.1 Network Diagram



This section will have the protocols and port numbers.

A little info on exactly what is required will be helpful?

Question: Do we need to show the database in the server as a separate object?

## 5.2 System Specification Summary

Docker containers - Unsure at the moment if it will all be in one or multiple containers.

## 5.3 Configuration Requirements

PostgreSQL: Port 5432

Django web server: Port 8000

Nginx?

# 

# 6 Solution Impact

This section should describe the impact which the solution has on processes, people and existing technology.

## 6.1 Impact on Processes

Cafes are places that usually use speakers to play music for their customers. The Virtual Jukebox application will change this process, by removing the need for speakers. The customer will be able to listen to the music that is hosted by the cafe, through their mobile phone, changing the usual process of playing music through a speaker.

## 6.2 Impact on People

When a person goes to a cafe or a music event they are only able to listen to the music through the speakers and they have no interaction with what is being played. The Virtual Jukebox application gives the users that join the session the ability to vote on what song is played next. This ability makes the user that is listening not only a consumer, but someone that can interact with a group.

## 6.3 Technical Impact

For the Virtual Jukebox application to work, the database will need to be stored on the amiritar server as well as the website being hosted on that server. This means that as long as the Virtual Jukebox application needs to work the server will need to be online, which uses electricity to run. The server will also need maintenance and if it is planned to be operational for more than 3-5 years then it is recommended to replace the servers[[How Often Should I Upgrade Servers? – alltasksIT](https://alltasks.com.au/often-upgrade-servers/)].

The Virtual Jukebox application requires a host to have a spotify premium account. This is due to the fact that the Virtual jukebox application uses the API from the spotify servers. The spotify servers need to be operational for the Virtual Jukebox application to work and as mentioned above the server will require electricity and maintenance.

To connect to the Virtual Jukebox application, a user will need to use a mobile or a laptop to be a host or a guest/listener. While connecting to the Virtual Jukebox application this will use a small percentage of the connected devices battery. If the connection is made using a cellular network, then there will also be data usage over the cellular network.

# 

# 

# 

# 7 Quality Attributes

In discussion of the quality attributes applicable to the solution, specific scenarios relating to each attribute, and in turn the solution, will be discussed with reference to a format defined as *Scenario-Investigation-Response* (SIR). Under this format, (i) an applicable scenario is presented, (ii) then the context of the scenario is identified, and lastly (ii) the appropriate response is outlined, along with its associated measure of verification.

## 7.1 Reliability

|  |  |
| --- | --- |
| **Scenario** | |
| Description | A fault in the Virtual Jukebox application causes a system-wide failure |
| **Investigation** | |
| Source | Internal to the application |
| Stimulus | A fault in the application |
| Artifact | The component in which the fault occurred |
| **Response** | |
| Action | The application handles any and all exceptions that may occur, allowing the system to fail gracefully |
| Measure | An error message detailing the resultant error, cause and location is logged |

## 7.2 Availability

|  |  |
| --- | --- |
| **Scenario** | |
| Description | The Virtual Jukebox is compromised by a Denial of Service (DoS) attack |
| **Investigation** | |
| Source | External to the application |
| Stimulus | The application receives more concurrent requests than it can handle |
| Artifact | The application |
| **Response** | |
| Action | The application blocks concurrent user requests, displaying a message to the user informing them that they need to wait momentarily in order to use a particular feature again |
| Measure | Length of attack: the DoS attack does not persist for any longer than 1 minute. |

## 7.3 Serviceability

|  |  |
| --- | --- |
| **Scenario** | |
| Description | Lack of documentation hinders implementation and testing |
| **Investigation** | |
| Source | Developers |
| Stimulus | Lack of prioritisation towards thoughtfully documenting system components, resulting in misrepresentation of system features and functionality |
| Artifact | Internal Virtual Jukebox application documentation |
| **Response** | |
| Action | Developers dedicate more time towards improving documentation |
| Measure | Time spent understanding the system that would have been aided by the provision of more robust documentation |

## 

|  |  |
| --- | --- |
| **Scenario** | |
| Description | The Virtual Jukebox application codebase has become to large and complex, making it difficult to add new features |
| **Investigation** | |
| Source | Developers |
| Stimulus | Designing and implementing the application without adherence to well-established design methodologies and patterns |
| Artifact | Software Architecture Specification |
| **Response** | |
| Action | Refactor code to be simpler and contain desired new functionality |
| Measure | Time spent refactoring code |

## 7.4 Best Practice Commentary

Performing technology investigation work prior to requirements. In essence, delivering the Technology Investigation Summary prior to the Software Requirements Specification, as having done the SRS first would have allowed us to more stringently pick alternatives based upon the degree to which they meet the requirements of the application

## 7.5 Architecture Commentary

If business requirements were to require that the application be monetized, Amristar would need to reach out to Spotify to obtain written permission allowing the monetisation of a product which uses their software.

# 8 Success Criteria and Test Plan

## 8.1 Use Case Validation

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | **Use Case Name** | **Validation** |
| MV-01 | Logging into a Virtual Jukebox Account | When the host clicks on the login button on the Virtual Jukebox application it will give bring them to the page where the user can enter their username and password.  If the password for the given username is correct it will notify the user that they have successfully logged in and bring the user to the homepage. |
| …. | …. | …. |

## 8.2 Validation of Solution metrics

Question: Does this section only need a table that shows the solution metrics?

Question: Do we do a Solution metrics for each use case or have it relating to the server and connected devices?

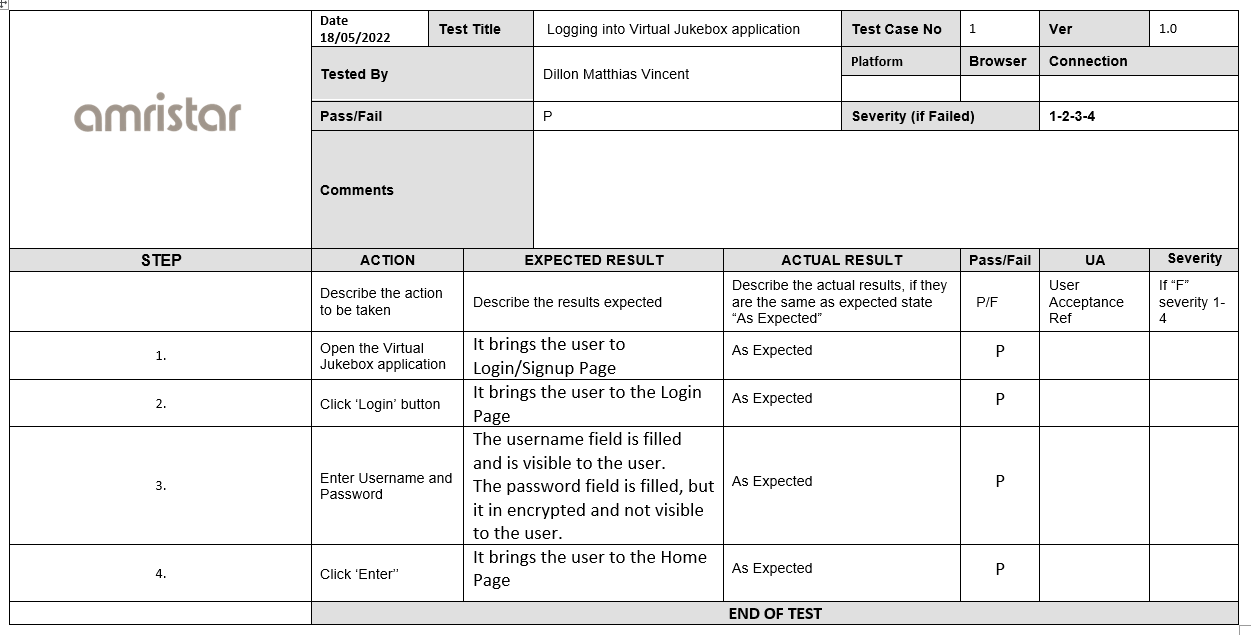
A little unclear on what to put down for the solution metrics.

|  |  |
| --- | --- |
| Application load time (first load) | 1 – 3 seconds |
| Application load time (subsequent loads, cached) | 1 – 2 seconds |
| Data refresh times | 1 – 2 seconds |

## 8.3 Test Plan

Question: Do we need to do a test plan for all the extensions for each use case?

Question: What do we need to put for Platform,Browser,Connection?



# 9. Phasing, Roadmap, and Implementation Plan

## 9.1 Technical Assumptions

## 9.2 Solution Phases and Technical Milestones

Will essentially provide a high level outline of the project plan

## 9.3 Technical Work BreakDown Structure

Will essentially translate 9.2 (the high level project plan) into a WBS

## 9.4 Implementation Instructions

Question: in the template it says “...It should not cover details already covered elsewhere, but should focus on specifics and “gotchas”...which are unlikely to be known to developers.” Any examples of what kind of gotchas we should be looking for?

Here, for each card of WBS, provide further details as to how to achieve that phase?

E.g. “4. Implementation -> 4.1. Coding”: Refer to the SRS for guidance in regards to the requirements of the application and prioritisation of various functionalities. Specifically, view sections 2.1 and 4.1”?

*and/or*

Here, perhaps make note of the different documents provided thus far, what they are and how they will assist in development?

e.g. TIS–detailed analysis of different candidate technologies for the project. Will provide assistance in letting developers know the fundamentals of developing with each given technology?

e.g. SRS–document detailing various requirements for the system as well as related components. Will provide assistance in helping developers prioritise the work which needs to be done, also accountability?

# 10 Appendix

# 11 References

# 

# 

# 

# 

# 

# 

# 