Use Cases and Logical Architecture

- XID: X00164430
- Name: Jakub Wysocki
- Project Title: Music Critique Website

Provide at least 6 Use-cases describing the functionality of the proposed system

Section 1: For Each Use Case:

Title (goal)	What is the name of this function?
Primary Actor	Who is the user?
Story	Describe in detail what happens here

Use Case 1:

Title (goal)	To become a registered user
Primary Actor	Non – Registered user
Story	**Successful Scenario:** 1. Non-Registered user enters the registration website 2. Non Registered user enters username 3. Non Registered user enters email 4. Non Registered user enters password 5. Non Registered user re-enters password 6. System verifies if username, email and passwords meet requirements 7. Non registered user becomes a registered user **Alternative Scenario:** 4. System notices requirements are not met, asks user to re-enter the invalid field

Use Case 2:

Title (goal)	To give a song a score
Primary Actor	Registered user
Story	**Successful Scenario:** 1. Registered user enters the post a review page 2. Registered user selects the rating they want to post 3. System validates if user is registered/logged in

	4. A song score is posted **Alternative Scenario:** 3. System notices user is not registered or logged in. Redirects to login/register page

Use Case 3:

Title (goal)	Post a comment
Primary Actor	Registered user
Story	**Successful Scenario:** 1. Registered user enters post a review page 2. Registered user types a comment that they want to leave 3. User submits the comment 4. System validates if the user is registered 5. Comment is submitted **Alternative Scenario:** 4. System notices user is not registered or logged in. Prompts user to login or register

Use Case 4:

Title (goal)	Send messages in the chatbox
Primary Actor	Registered user
Story	**Successful Scenario:** 1. Registered user enters online chat screen 2. Registered user types message in the chat box 3. Registered user clicks send 4. System Validates if user is registered 5. Message is displayed in the online chat **Alternative Scenario:** 5. System notices user is not registered or logged in. Doesn't allow user to type a message. Prompts user to login or register

Use Case 5:

Title (goal)	Modify or delete existing comments
Primary Actor	Website admin/ moderator
Story	**Successful Scenario:** 1. Website admin enters post a review page 2. System validates if the user is an admin 3. Admin gets access to edit tools to moderate the website

	4. Admin clicks trash icon to delete a comment 5. Comment Is deleted **Alternative Scenario:** 2. System detects user is not an admin. Doesn't display edit tools

Use Case 6:

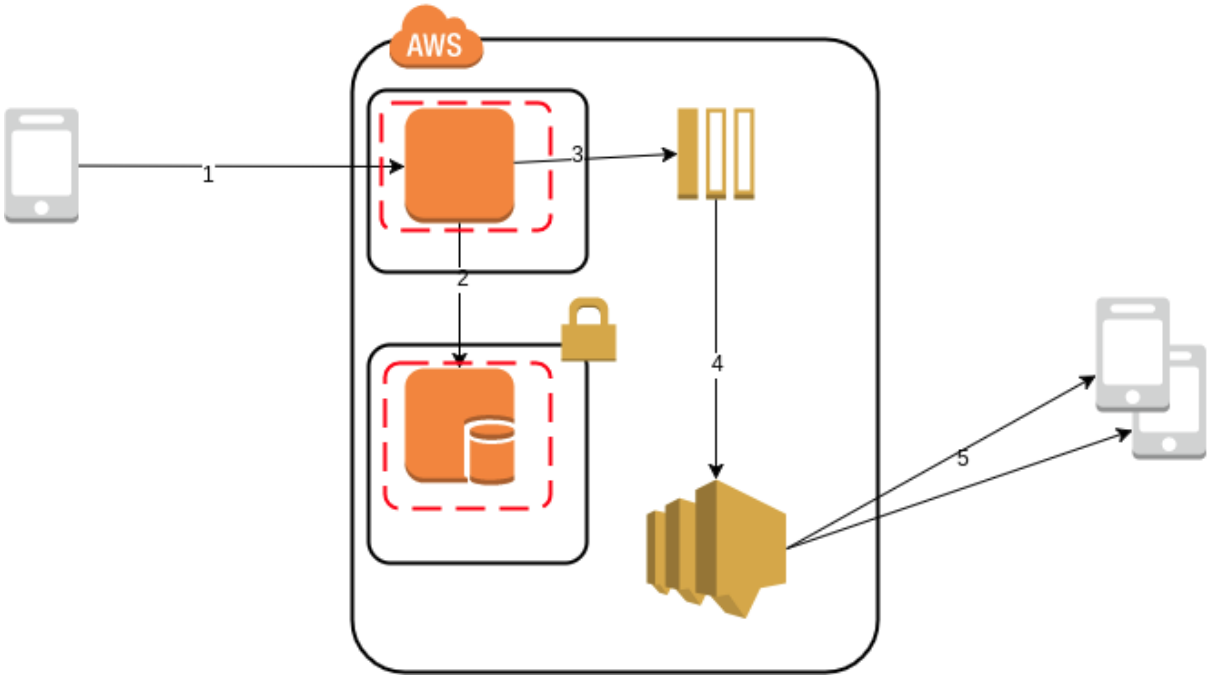
Title (goal)	To generate Spotify report
Primary Actor	Spotify Authorized User
Story	**Successful Scenario:** 1. User authorises login with Spotify 2. API pulls down users Spotify profile 3. A report is generated with details about users profile such as favorite artist and songs 4. Report is displayed

Section 2: Logical Architecture

In this section use www.draw.io to show how your architecture is distributed. Your architecture should to show:

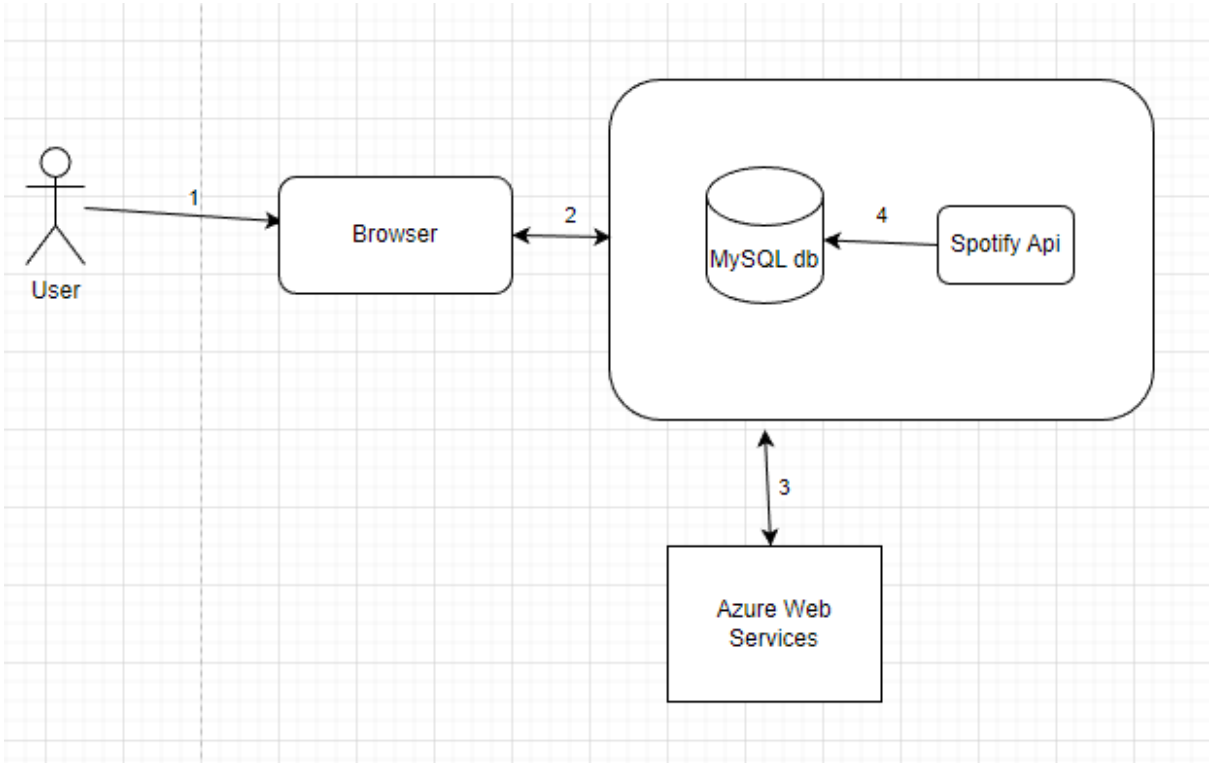
Software Components
Databases
App engines
Mobile platforms
APIs and languages (per component)
Deployment EG RESTful, JDBC, sessionless etc
Security e.g. Https, certificates, authentication, etc.

Here is an AWS sample architecture:



Logical Architecture Discussion

- Discuss each component of your architecture
- Add numbers to the arrows and discuss the flow of data



The browser is the front end of the system. It retrieves, displays and submits information given by the user to the back-end system.
The database stores data both of the user and the requested Spotify API
The Spotify API is used to take charting songs from Spotify servers
Azure web services is where the system will be deployed and hosted.

Flow of data
The user connects to the front end of the system using a browser. Through the browser they receive and send information to and from the back end where the database is located. The database communicates with the Spotify API to retrieve information from the Spotify servers. All this is done on azure web services that provide the tools to deploy and host the system.