Analysis Document

S6 Software Engineering

4207734

Matei-Cristian Mitran

Fontys Eindhoven

08.03.2023

Introduction

YouSound is an enterprise-grade university project, developed individually in the sixth semester. It is a fully featured music platform that aims to provide users with a unique music listening experience. The project incorporates all the essential features of a standard music platform, including a user-friendly interface, playlist creation, music sharing, and discovery tools.

Functional Requirements

1. The system will allow users to stream music.
2. The system will allow users to search music-related content.
3. The system will allow users to interact with other users.
4. The system will allow users to post messages and read messages in artist's communities.
5. The system will allow administrator users to moderate music-related content.
6. The system will allow administrator users to moderate the platform by managing users.
7. The system will allow artist users to post music.
8. The system will allow artist users to create and build their communities.
9. The system will allow artist users to update their music.
10. The system will allow artist users to interact with their audience.

User Stories

Format

- *[User Story]*
  - *[Acceptance Criteria 1]*
  - *[Acceptance Criteria 2]*
  - *[Acceptance Criteria 3]*
- *[Content] = Any content that can be uploaded to the platform. Music, Albums, Podcasts, Posts.*

- As a user, I want to create a new account on the platform.
  - User can log in to the platform.
  - User is stored in the database safely.
  - Data provided by user is validated.
- As a user, I want to search for music-related content.
  - User can see all the content with a similar title as the search input.
  - User can interact with the displayed content.
  - Search results are modified anytime the input changes.
- As a user, I want to join an artist's community.
  - User can see information about the community (members, posts, songs).
  - User successfully is added to the community user list.
  - User can successfully post or interact with content in the community.
- As a user, I want to create a playlist.
  - User can see his playlist in the sidebar.
  - User can interact and configure his playlist.
  - Playlist is successfully stored in the database.
- As an admin user, I want to manage a user.
  - Admin can successfully ban or update a user.
  - Admin can send a message to a user.
  - Admin can see relevant information about the user.
- As an admin user, I want to manage content.
  - Admin can successfully remove, update, flag content.

- As an artist user, I want to stream my music.
    - Artist can successfully upload music.
    - Artist can successfully interact with own music.
    - Music is safely stored in the database.
- As an artist user, I want to create a community.
    - Artist can successfully manage their communities.
    - Artist receives notifications from the community.
    - Community is safely stored in the database.
- As an artist user, I want to send a private message to a listener.
    - Artist can successfully send a message.
    - Listener can see the message.
    - Message is safely stored in the database.

Non-Functional Requirements

Non-Functional requirements are very important to a software engineering project. They specify how a system should perform and interact with users while ensuring its quality, reliability, and suitability for its scope. To develop an enterprise-grade application, YouSound should include the following non-functional requirements:

1. **Performance**: The application should be fast to respond, load and interact with.
    a. *Response Time:* 200-500 ms
    b. Throughput: 5000 transactions per second
    c. CPU Utilization: below 70% CPU

2. **Scalability**: The application should be able to handle a large number of users.
    a. Test horizontal scalability: Scale out to 10000 transactions
    b. Test vertical scalability.
    c. Elasticity: Scale up or down within 5 minutes

3. **Reliability**: The application should be always available, and the downtime reduced as much as possible to ensure an enterprise grade service.
    a. Measure time between system failures: 500 hours ideally
    b. Measure time of system fixes: 1-3 hours ideally

4. **Security**: The application should be secure and not vulnerable to attacks, protecting user data from unauthorized entities.
    a. *Test authentication and authorization*
    b. *Test encryption by attempting to decrypt data without authorization.*

5. **Architecture**: The application's architecture should be designed keeping in mind all the other non-functional requirements that ensure an enterprise-grade application.
    a. *Testability: create tests for the components that make up the system*
    b. Modularity: test to see if system works correctly if you remove/add components

6. **Maintainability**: The application should be easy to update and configure, ensuring maintainability.
    a. Transferability: documentation
    b. Code maintainability: quality assurance with SonarQube

7. **Compatibility**: The application should be compatible with different devices and browsers.
    a. *Browser: the application should be compatible with different browsers*
    b. *Operating System: the application should function on different OS*
    c. *Device: the application should run as expected on different devices*

8. **Usability**: The application should provide a good user experience, with an easy-to-use interface and inviting design.
    a. *UI Testing*

Misuse Cases

- **MUC-01**: User uploads copyrighted music without permission.

  1. User uploads a song that is owned by someone else without obtaining the necessary permissions or licenses.
  2. User intentionally or unintentionally uploads music that they do not have the rights to, which could result in legal action against the company.

  Solution: Implement a process for admins to approve music-related content on the platform.

- **MUC-02**: User creates a fake artist account.

  1. User creates an account pretending to be a well-known artist in order to deceive listeners and gain exposure.
  2. User uses the fake artist account to upload their own music, which is against the platform's terms of use.

  Solution: Implement a verification process for artist accounts to ensure that they are legitimate. This could include requiring artists to provide proof of their identity or requiring them to verify their account through a third-party service. Additionally, monitor for suspicious activity and act against any users found to be creating fake accounts.

- **MUC-03**: User engages in spamming or phishing activities.

  1. User sends unsolicited messages or posts spam content on the platform.
  2. User sends phishing messages in an attempt to obtain sensitive information from other users.

  Solution: Implement measures to detect and prevent spam and phishing activity, such as requiring users to verify their account before sending messages or implementing spam filters to block messages containing certain keywords. Provide users with clear guidelines for appropriate use of the platform and consequences for violating those guidelines.

- **MUC-04**: Admin abuses their privileges.

    1. Admin uses their access to user data for personal gain or to harm other users.
    2. Admin deletes content or accounts without proper justification or due process.

    Solution: Thoroughly verify administrators before creating an account for them to avoid any abuse of privileges.

- **MUC-05**: User engages in hate speech or harassment.

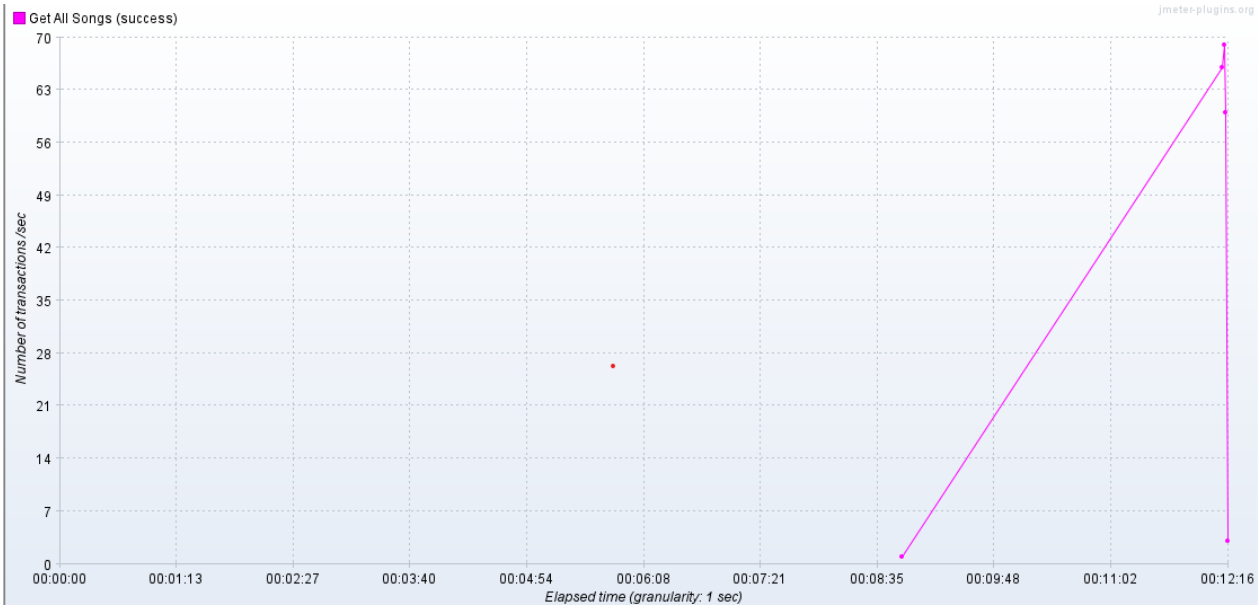
    1. User posts content or comments that contain hate speech, derogatory language, or threats.
    2. User repeatedly harasses other users through private messages or comments.

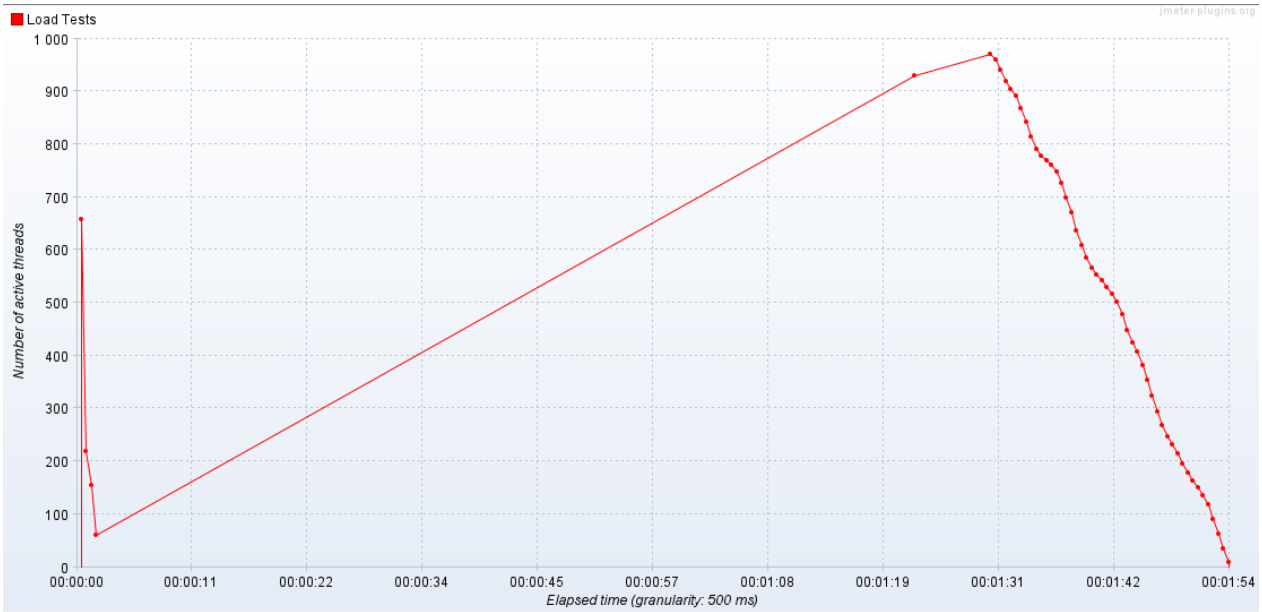
    Solution: Implement moderation policies to detect and remove hate speech and harassing content. This could include using AI tools to automatically flag content containing offensive language or images, and then having a human moderator review and take appropriate action. Allow users to report harassment or hate speech and investigate reports promptly.

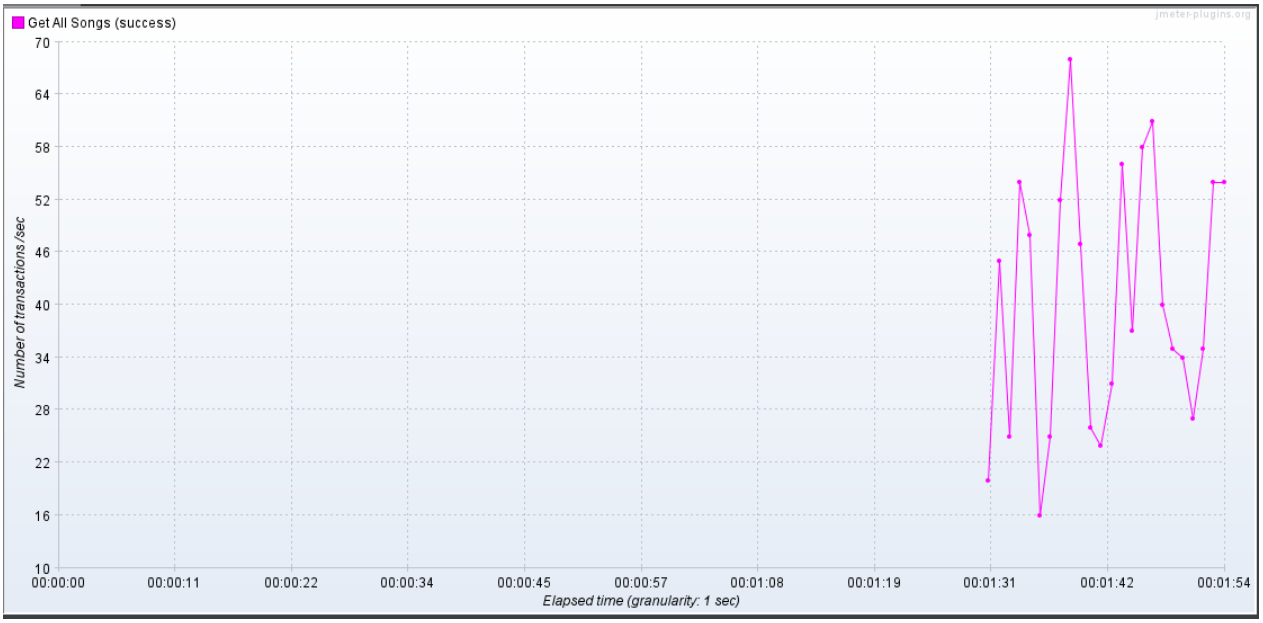
Performance Tests Results

Load Tests

I simulated a load of 2500 users at once and performed tests on the endpoints that would have most value, get song by id and get songs to find out how the system performs on peak load. These are the results that validate the performance non functional requirements showing a maximum of 70 transactions per second.

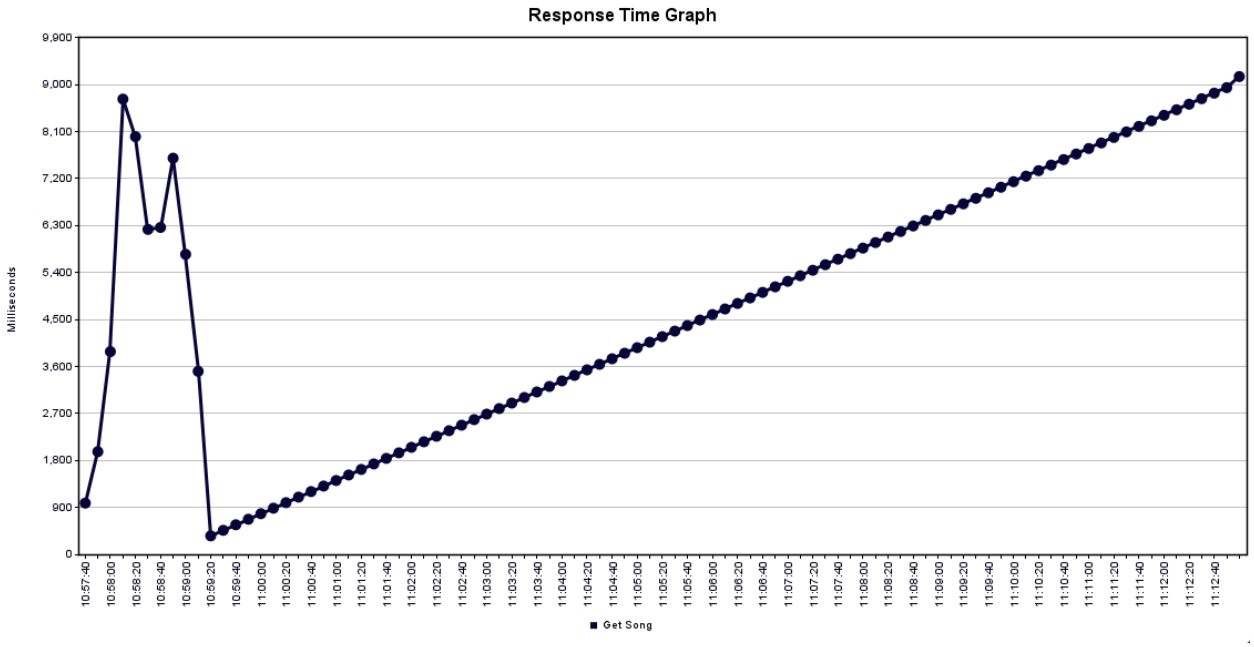






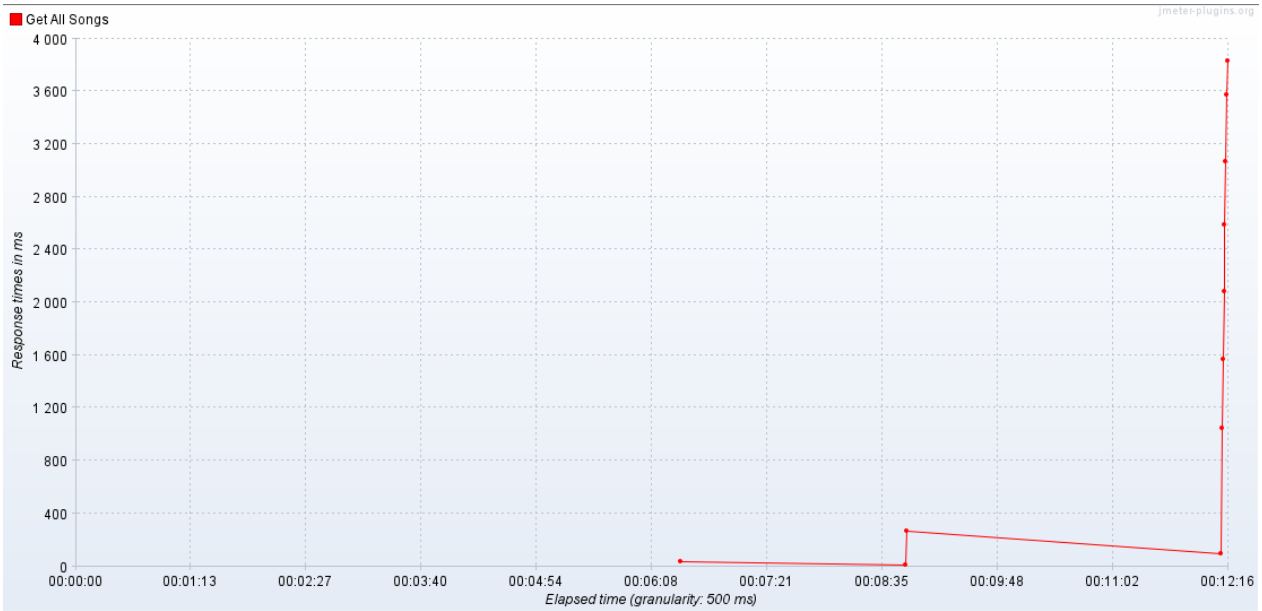
Stress Tests

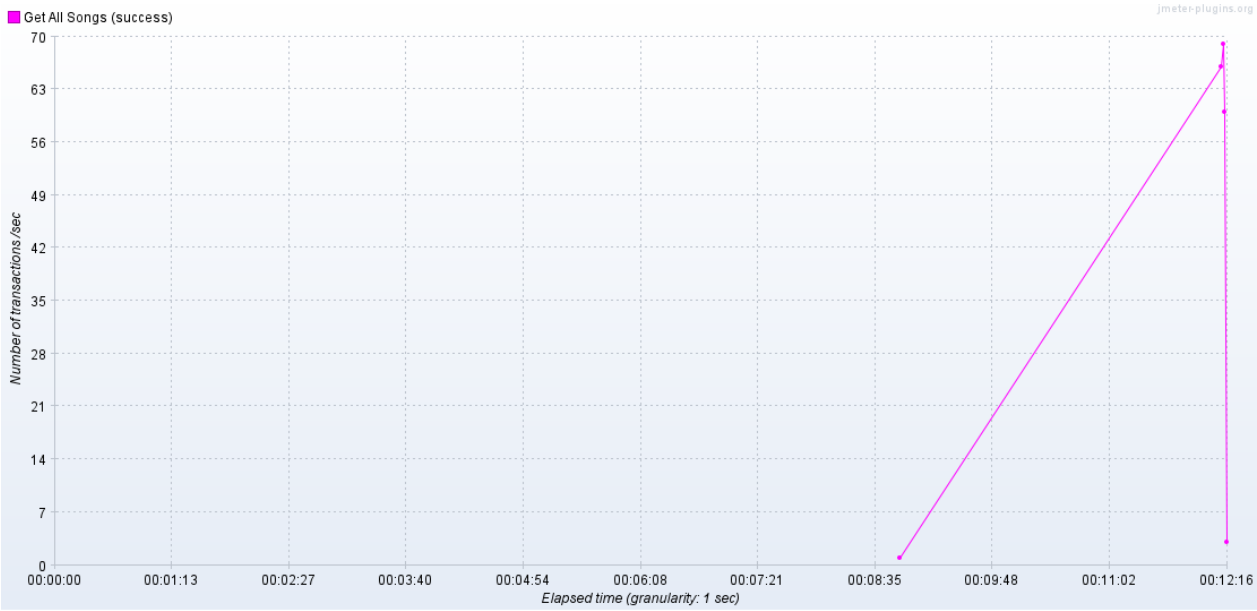
I performed stress tests to find out how the system performs on a load of 100 users, then increase it to 1000 and then to 2500. These are the results.



Spike Tests

I performed spike tests to find out how the system would perform if the load were increased at once very much, to a total of 3500 users. These are the results:




Get All Songs (success)
jmeter-plugins.org
70
63
56
49
42
35
28
21
14
7
0
Number of transactions /sec
00:00:00
00:01:13
00:02:27
00:03:40
00:04:54
00:06:08
00:07:21
00:08:35
00:09:48
00:11:02
00:12:16
Elapsed time (granularity: 1 sec)


Get Song
jmeter-plugins.org
1 000
900
800
700
600
500
400
300
200
100
0
Response times in ms
00:00:00
00:00:06
00:00:13
00:00:20
00:00:27
00:00:34
00:00:41
00:00:48
00:00:55
00:01:02
00:01:09
Elapsed time (granularity: 500 ms)