Trifonov,Dobri D.V.

Test PLan

Contents

[Testing Strategy 2](#_Toc88852716)

[Context 2](#_Toc88852717)

[Front-End 2](#_Toc88852718)

[Back-End 2](#_Toc88852719)

[Database 2](#_Toc88852720)

[Testing 3](#_Toc88852721)

[Front-End Testing: Cypress 3](#_Toc88852722)

[Back-End Testing: Unit tests and SonarQube 3](#_Toc88852723)

[Database Testing 3](#_Toc88852724)

|  |  |  |
| --- | --- | --- |
| **Version** | **Description** | **Date** |
| 1.0 | Creation of test plan document | *26/11/****2021*** |
| 1.1 | Added Test Plan | *17/12/2021* |

# Testing Strategy

## Context

To create a test plan first we need to understand the components of our application

### Front-End

The front end uses React, an open source library, that is used for building single page applications and is compatible with both web and mobile apps. React consists of HTML, CSS and JavaScript. In the project we also implement some libraries/modules that help to provide needed functionality.

### Back-End

The back end uses Spring boot, a Spring framework that is optimized for shortening the code length and providing an easy way to run a Spring application. At it’s core it is Java that is heavily reliant on microservices.

### Database

For the database we use MySQL since it is one of the most popular and reliable SQL databases.

# Testing Software

## Front-End Testing: Cypress

For the Front end we have decided to use Cypress, since it allows to write automated web tests in JavaScript and has a relatively simple set up process.

## Back-End Testing: Unit tests and SonarQube

The nature of the application makes it difficult to implement Unit testing, since most of it consists of passing objects and variables around, but we will attempt our best to reach 40% code coverage and if possible go for 60%. Even tough there isn’t that much logic, unit tests will ensure that code behaviour remains the same through the application’s life cycle.

SonarQube will also be implemented to ensure the code quality remains. Since SonarQube can detect small bugs and security concerns it will be vital for developing a bug free application.

Diagram

Description automatically generated with medium confidence

## Database Testing

There really isn’t a reason to test the database. It’s a tried and tested product that has plethora of inbuilt checks to ensure that data remains consistent.

# Testing

## Unit tests

Unit tests will be the core of our testing strategy. Most of the code will be covered by them. They can be ran in the middle of development and can immediately detect code that is behaving irregularly.

## Integration tests

Integration tests are integrated into the CI pipeline of the git project. The application will be built, the unit tests will be ran and finally Sonar-Cloud will evaluate the code quality. This will automate the process of checking if our unit tests are working since we wont have to manually build our application.

## User Acceptance Tests

The User Acceptance Tests require great time and effort to create, so there will be very few of them. The application will be given to random people and all feedback would be recorded.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Scenario | Requirements | Acceptance Criteria | Testing plan |
| 1 | As a user, I want to create an account | User has a valid email address | User can create an account | - Tester open the registration page.  - Provide valid information |
| 2 | As a user, I want to login into my account | User has already registered | User can login | -Tester click on the login button  -Tester provides valid credentials |
| 3 | As a user, I want to be able to logout of my account | User is logged in. | User can logout | -Tester clicks on the logout button in the header. |
| 4 | As a user, I want to view all of my playlists | Collection of playlists in user’s account | User can see all of his available playlists | -Tester opens the playlist off canvas |
| 5 | As a user, I want to create a playlist to my account | Playlist added from the front end | User can add a playlist | -Tester opens the playlist off canvas.  -Tester selects the add button  -Tester provides a name for the playlist |
| 6 | As a user, I want to delete a playlist in my account | User has already added a single playlist | User can delete a playlist | -Tester opens the playlist off canvas.  -Tester clicks on the delete button adjacent to the playlist that is to be deleted  -Tester accepts the delete confirmation popup |
| 7 | As a user, I want to add music to my account | User is logged in.  User has a link to a video source | User can add media to their account | -Tester opens the playlist off canvas  -Tester click the song button adjacent to the playlist  -Tester provides valid required information |
| 8 | As a user, I want to remove music from my account | User is logged in.  User has a song in his account | User’s selected song is deleted | -Testers selects the song that is to be deleted  -Testers accepts the delete confirmation popup |
| 9 | As a user, I want to play music from the application | User is logged in.  User has added a playlist.  User has added a song | User can play a media file | -Tester clicks on the music player button in the header.  -Tester clicks on the play button. |
| 10 | As a user, I want to navigate through my playlist | User is logged in.  User has added a playlist.  User has added a songs | User can select a specific song | -Tester navigates to the music player  -Tester pressed the skip forward button.  -Tester pressed the go backwards button. |
| 11 | As a user, I want to adjust the volume of my music | User is logged in.  User has added a playlist.  User has added a songs | User can change the volume of media | -Tester adjust the volume slider in player |
| 12 | As a user, I want to change position of my music | User is logged in.  User has added a playlist.  User has added a songs | User can change the location of the playing media. | -Tester adjust the progress slider in player |
| 13 | As a user, I want to see my profile | User is logged in. | User can have a personalized account | -Tester click on the name adjacent to the “Signed in as:” label |