S28299 Mariia Okunieva Project documentation

1. Application name:
LingoLyric

2. Problem area:
The application is designed to improve foreign language skills via learning song lyrics. It serves both as educational and entertainment tool.

3. Project goals:
LingoLyric was created to combine the love of music with language learning. The idea for this project emerged from my personal need as someone who loves Italian music but also wants to understand the lyrics and learn the language more effectively. By integrating music videos, lyrics, and translations, LingoLyric aims to make language learning engaging and enjoyable. This system facilitates learning by leveraging the emotional connection and repetition inherent in music, making it easier for users to memorize and understand new languages.

4. System responsibilities:
The system enables users to register, select a language, system shows songs based on a selected language, allows to search for songs, view lyrics and translations and listen a song while learning, take quizzes, and track the progress.

5. System users:
- Visitor
- User

6. Functionalities:
- Visitor registration and login:
  Visitors can create an account and log in to become a user and access application functionalities. When registering one also specifies a language they wish to learn.
- Language selection:
  Users change the language they would like to learn any time.
- View songs available:
  The system displays songs available in the selected language.
- Search for songs:
  Users can search for songs by entering the title or author of song.
- Add songs to favourites :
  Users can add songs to favourites list for quick access.
- Selection of a song:

Users can select a song from the list to begin the learning process.

- Streaming a music video:
  Users can watch the selected song's music video streamed from YouTube.
- Lyrics and translation display:
  The system displays the songs lyrics along with English translation.
- Take quizzes:
  Users can take quizzes designed around the lyrics of the selected song.
- Learning process tracking:
  After completing a quiz the system adds to a song a label indicating the user's level of mastery of the song in percentages and adds to the list of user's progress.
- View list of progress:
  Users can access the list of progress for all the quizzes they took.

7. User requirements:

Part one:

Entities and properties:

- User
  - id (unique user id)
  - username (username specified by user)
  - password (password for log in)
  - email (email used to log in)
  - language (selected language for learning)
- Song
  - id (unique song id)
  - title (title of the song)
  - artist (name of song's artist)
  - language (language of the song)
  - lyrics (lyrics of the song)
  - translation (English translation of lyrics)
  - videoUrl (URL of the YouTube video)
- Quiz
  - id (unique quiz id)
  - songId (id of a song related to quiz)
- QuizResult
  - id (unique id for each quiz result)
  - userId (id of a user who took a quiz)
  - quizId (id of a quiz taken)
  - score (score obtained by user in quiz)
  - date (date when the quiz was completed)
- QuizQuestion
  - id (unique id for each quiz question)
  - quizId (id of the quiz to which question belongs)

  - question (question text)
  - correctAnswer (text of correct answer)
- Favourite
  - userId (id of user who added the song to favourites)
  - songId (id of the added song)
- Progress
  - userId (id of user)
  - songId (id of a song)
  - masteryLevel (mastery level in percentage)

Relationships between entities

- User to Favourites: One-to-Many relationship, where a user can have multiple favourite songs.
- User to Progress: One-to-Many relationship, where a user has mastery levels for multiple songs.
- User to QuizResult: One-to-Many relationship, where each user can have quiz results for multiple songs.
- Song to Quiz: One-to-One relationship, where each song has a related quiz.
- Quiz to QuizQuestion: One-to-Many relationship, where a quiz has multiple questions.
- Quiz to QuizResult: One-to-Many relationship, where a quiz can have the results of multiple users.

Part two:
Expected functionality:
Visitor:

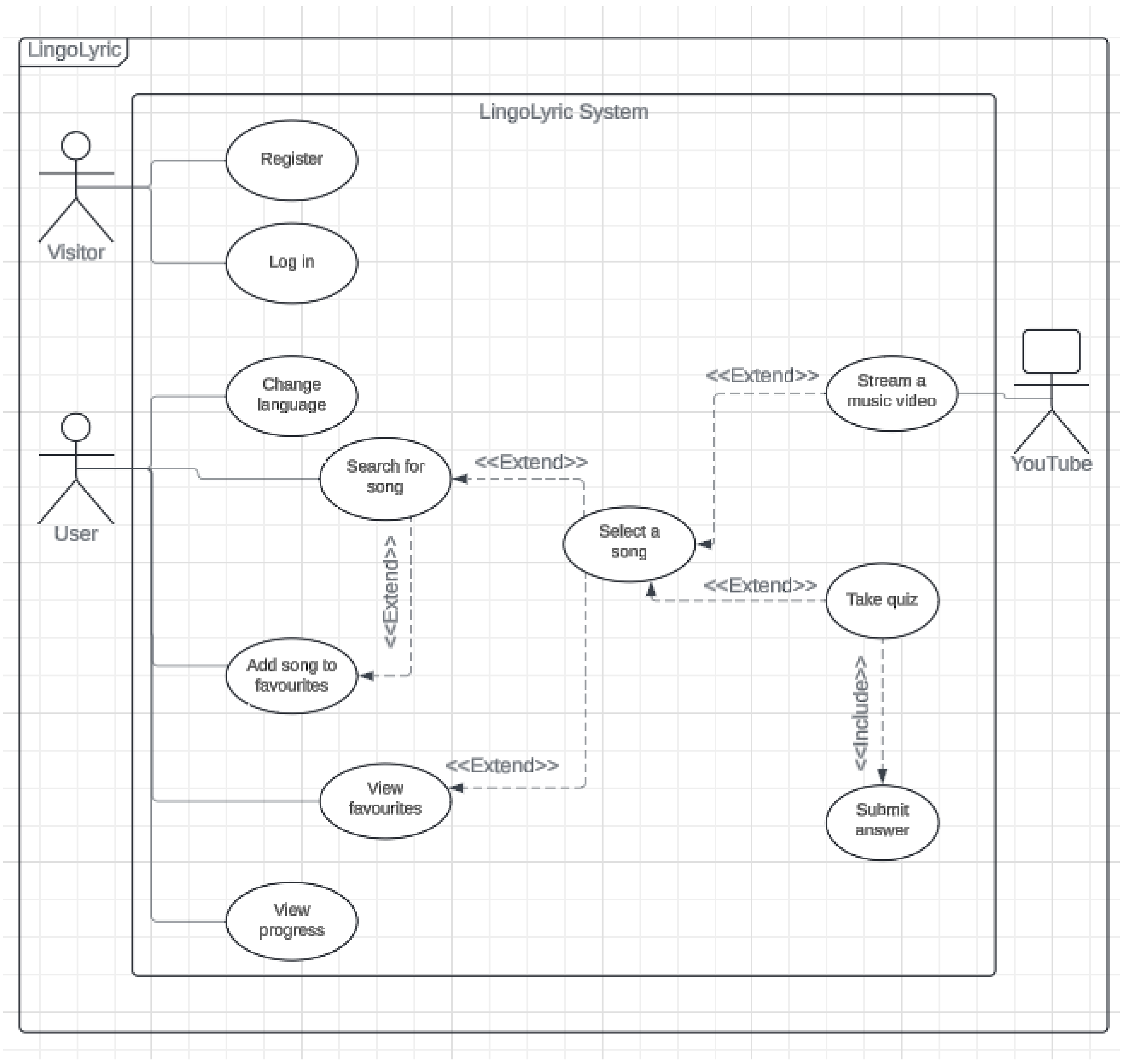
- Visitors can create an account or log in to become a user and access personalized features. (Methods: register(), login())

User:

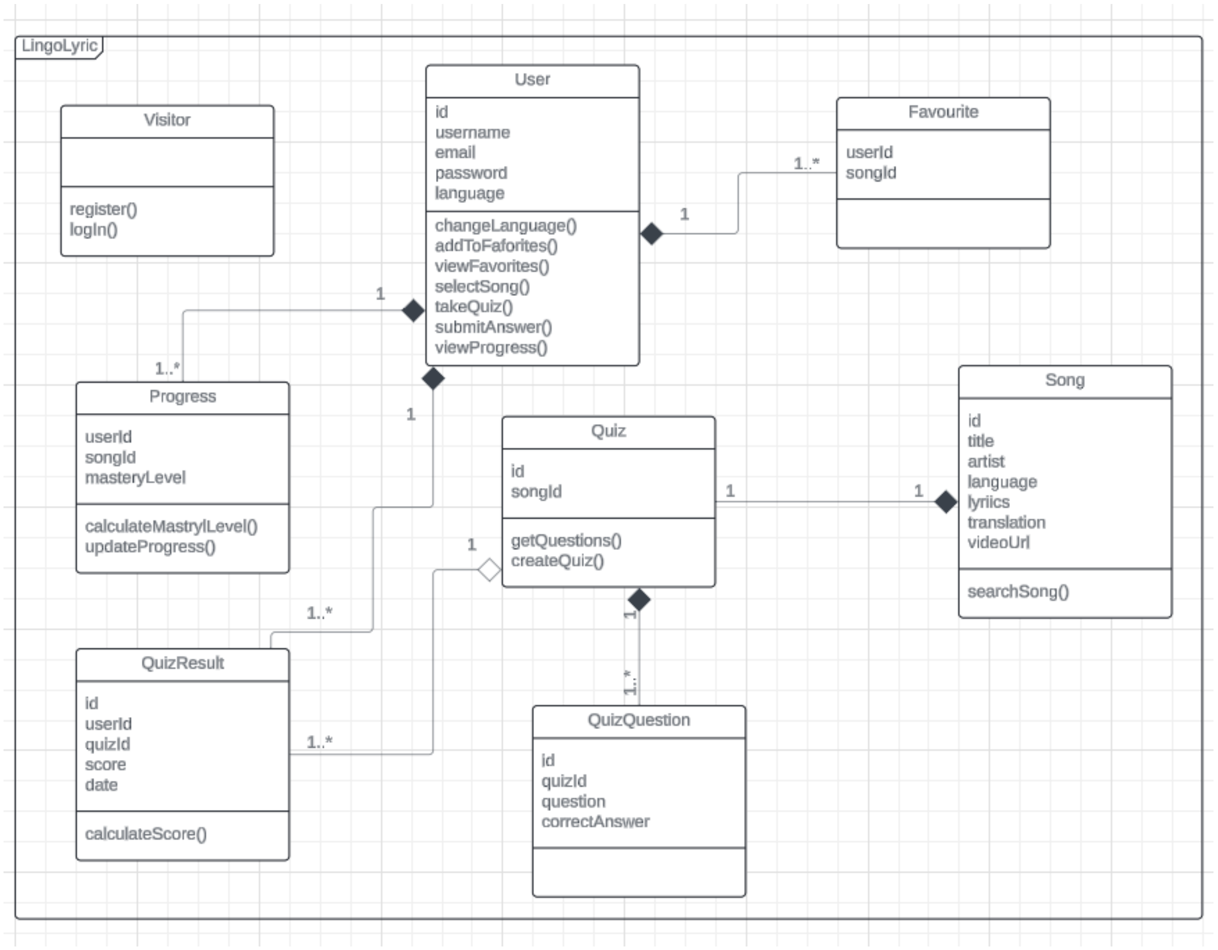
- Users can change the language they wish to learn. (Method: changeLanguage())
- Users can search for songs by entering the title or artist's name. (Method: searchSongs())
- Users can add songs to favourites list. (Methods: addToFavourites(), viewFavourites())
- Users can select a song from the list to begin the learning process. (Method: selectSong())
- User can turn on the music video of a song streaming. (Method: startVideoStreaming()) External system (YouTube) handles the video playback and streaming.
- Users can take quizzes based on a song lyrics. (Methods: takeQuiz(), submitAnswer())
- Users can access the list of their progress from the quizzes they took. (Method: viewProgress())

Part three:
Constraints:
1. Specific environment: The application will be developed and tested in a local environment, running on local host.
2. Expected reliability: The system should be stable and able to handle typical user interactions without crashing or significant bugs.
3. Performance: The system should be responsive, with page load times of less than 2 seconds under normal usage conditions.
4. Usability: The application should have an intuitive and user-friendly interface, ensuring that users can easily navigate and use the system.
5. Scalability: The system should be designed in a way that it can be scaled up in the future, but for now, it will only support a limited number of hardcoded songs.
6. Data Integrity: Ensure that the data remains consistent and accurate during interactions.

8. Functional requirements:



9. Description of the system structure:



10. Non-functional requirements:

1. Specific environment

Constraint: The application should run on a local environment.

Metric: The application must be accessible and fully functional on localhost (http://localhost:8080).

2. Expected reliability

Constraint: The system should not crash and should handle user interactions smoothly.

Metric: During a 1-hour test session, the application and should handle at least 20 different user interactions without errors.

3. Performance:

Constraint: The system should be responsive, with quick page load times.

Metric: Page load times should be less than 2 seconds for all main functionalities.

4. Usability:

Constraint: The application should be easy to use and navigate.

Metric: Users should be able to perform basic tasks (e.g., register, log in, select a song) within 2 minutes without needing additional help or instructions.

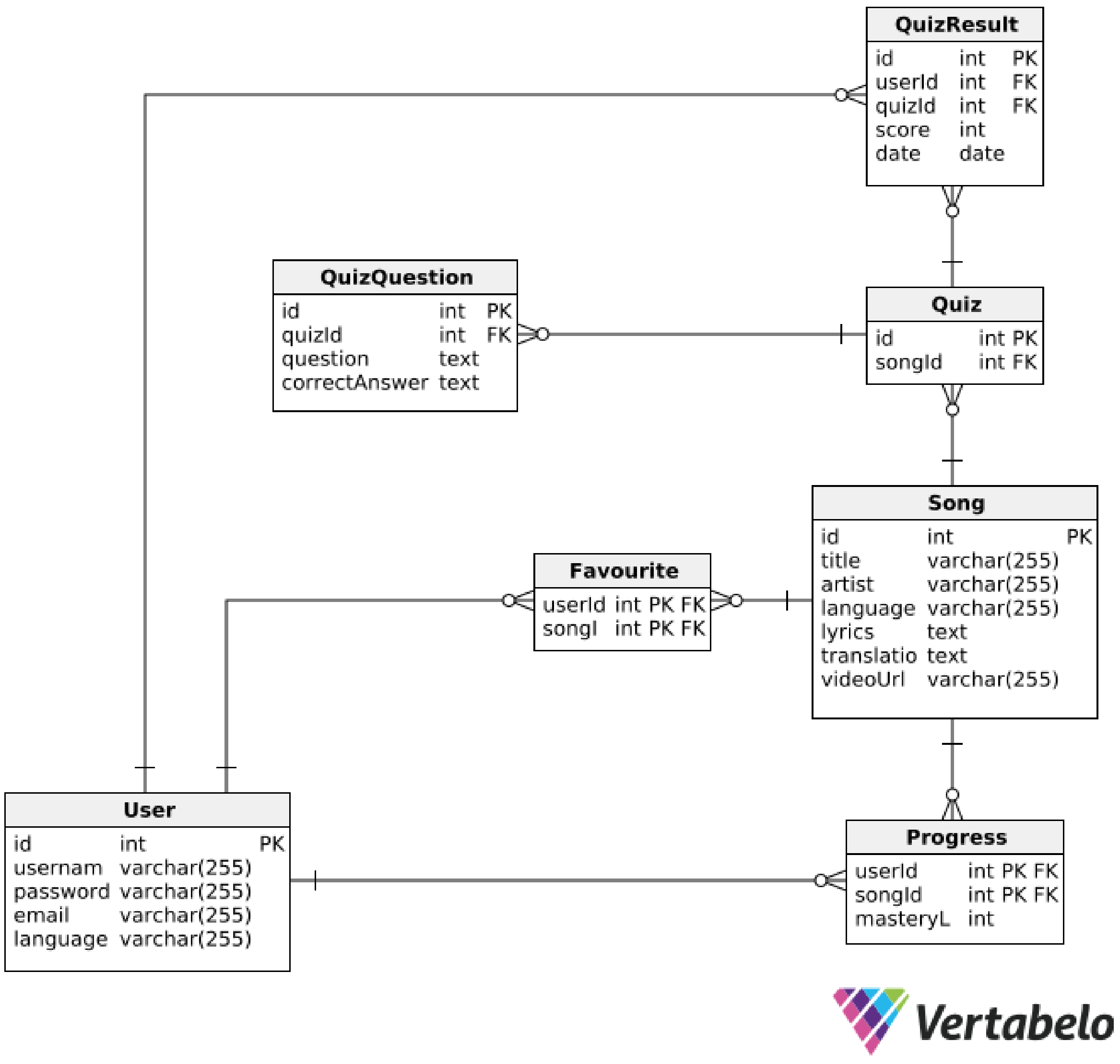
5. Scalability:

Constraint: The system should be designed in a way that allow future scalability.
Metric: The application architecture should support adding more songs and with minimal changes to the codebase.

6. Data integrity:

Constraint: Data must remain consistent and accurate.
Metric: No data corruption or loss should occur during typical operations, and data should be validated before being processed or stored.

11. Database schema:



12. API endpoints list with description:

1. Register
   - Endpoint: POST /api/lingolyric/visitors/register

- Description: Register a new user by providing username, password, email and language.
- Request body:
  ```
  {
    "username": "string",
    "password": "string",
    "email": "string",
    "language": "string"
  }
  ```
- Response: Redirects to the main page with songs available in chosen language or provides error details.

2. Login
   - Endpoint: POST /api/lingolyric/visitors/login
   - Description: Log in with username and password to access the user's personalized account.
   - Request body:
     ```
     {
       "username": "string",
       "password": "string"
     }
     ```
   - Response: Redirects to the main page with songs available in chosen language or provides error details.
3. Change language
   - Endpoint: PUT /api/lingolyric/changeLanguage
   - Description: Change the language the user is learning.
   - Request parameter:
     language (the new language to set)
   - Response: Redirects to the main page with songs available in chosen language or provides error details.
4. Search song
   - Endpoint: GET /api/lingolyric/songs/search
   - Description: Search for songs by title or artist's name.
   - Request parameter:
     query (the search query string)
   - Response: List of songs matching the search criteria.
5. Add to favourites
   - Endpoint: POST /api/lingolyric/users/favourites
   - Description: Add a song to the user's favourites list.
   - Request parameter:
     songId (the id of a song to be added)
   - Response: Success message or error details.

6. View favourites
    - Endpoint: GET /api/lingolyric/users/favourites
    - Description: View the user's list of favourite songs.
    - Response: List of favourite songs.

7. Select song
    - Endpoint: GET /api/lingolyric/songs/{songId}
    - Description: Select a song to begin learning process.
    - Path parameter:
      songId (the id of the song to select)
    - Response: Details of the selected song.

8. Take quiz
    - Endpoint: GET /api/lingolyric/quizzes/{quizId}/questions/{questionNumber}
    - Description: Get the specified quiz question
    - Path parameters:
      quizId (the id of the quiz)
      questionNumber (the number of the question to retrieve)
    - Response: Details of the specified question.

9. Submit answer
    - Endpoint: POST /api/lingolyric/quizzes/{quizId}/questions/{questionNumber}
    - Description: Submit an answer to the specified question and get the next question or the final result.
    - Path parameters:
      quizId (the id of the quiz)
      questionNumber (the number of the question to retrieve)
    - Request Body:
      ```
      {
        "answer": "string"
      }
      ```
    - Response: Next question details or final result with score.

10. View progress
    - Endpoint: GET /api/lingolyric/users/progress
    - Description: View the user's progress on different songs.
    - Response: List of progress records.

13. **List of used technologies**:

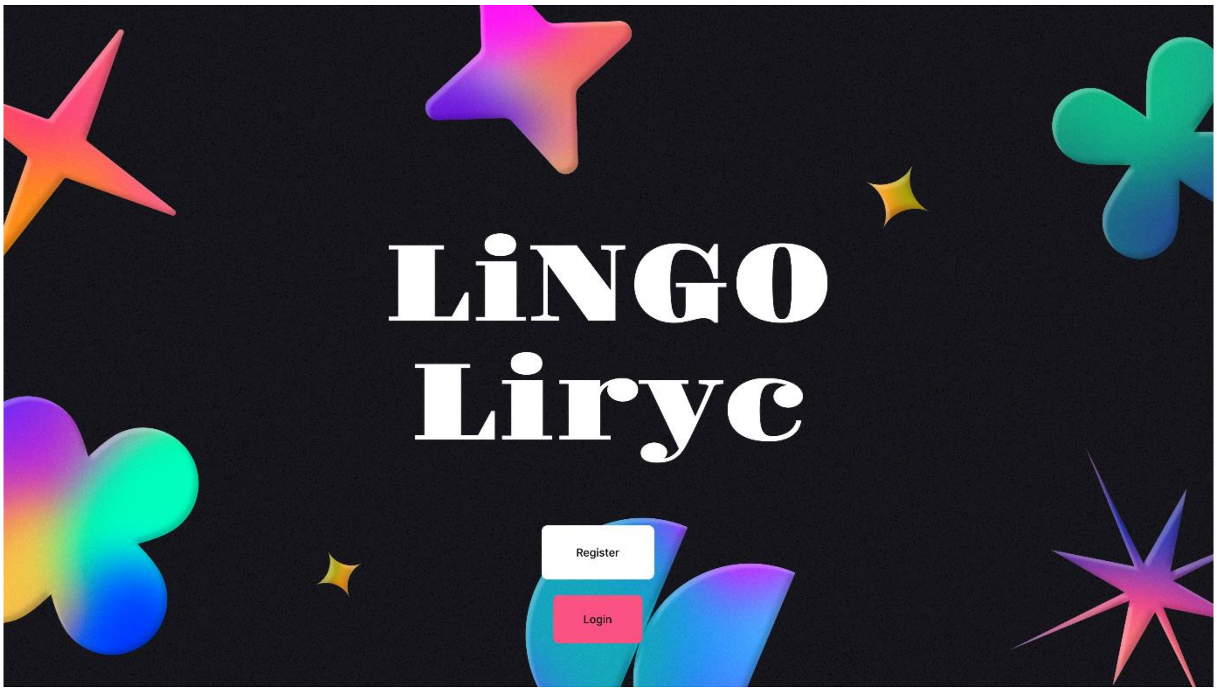
1. Java
    - Purpose: Primary programming language for developing backend.
    - Usage: Implementing core application logic, defining entity models, creating RESTful API endpoints.

2. Spring Boot
    - Purpose: Framework for building and running the backend server.
    - Usage: Creating RESTful services and handling requests.

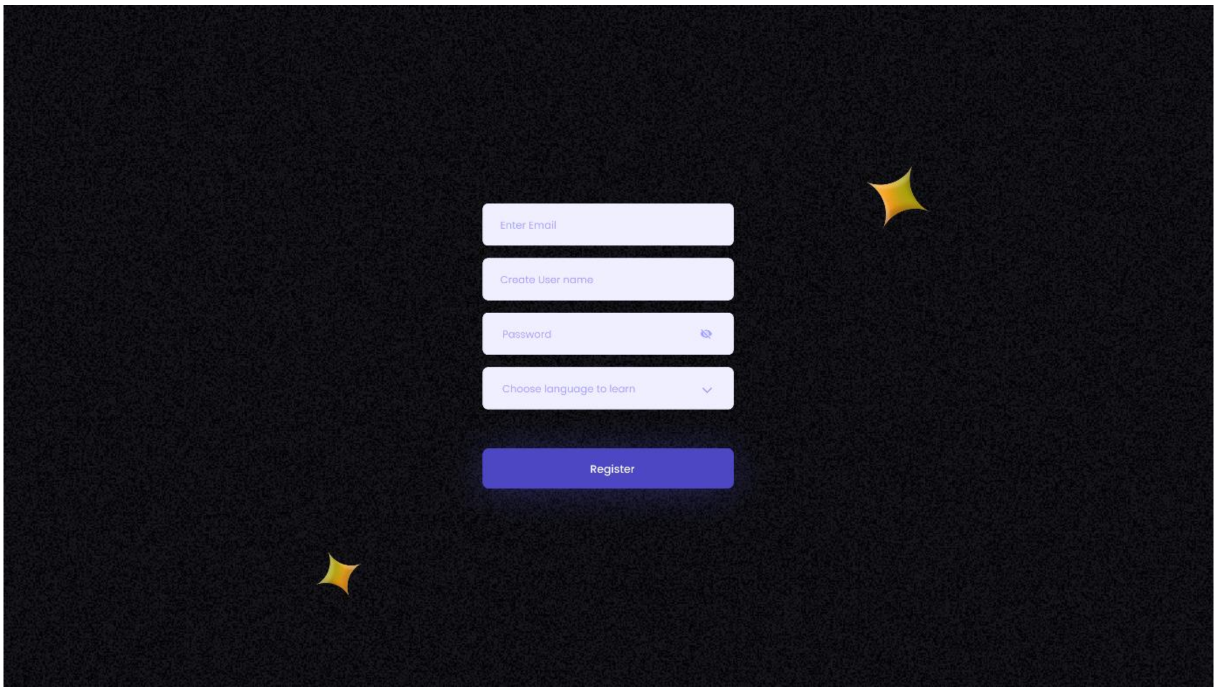
3. Spring Data JPA
    - Purpose: Simplifying database interactions.
    - Usage: Defining repositories for data access, implementing CRUD operations, and mapping Java objects to database tables.
4. Spring Security
    - Purpose: Securing the application.
    - Usage: Implementing user authentication and authorization.
5. MySQL
    - Purpose: Database for persistent storage.
    - Usage: Storing user information, song data, quiz questions, quiz results.
6. Gradle
    - Purpose: Dependency management and build automation.
    - Usage: Managing projects dependencies, building a project.
7. Thymeleaf
    - Purpose: Server-side template engine for rendering HTML.
    - Usage: Creating dynamic web pages by integrating data from backend with HTML templates.
8. Bootstrap
    - Purpose: Frontend framework for responsive design.
    - Usage: Styling and layout of web pages, ensuring the application is different device-friendly and visually appealing.
9. HTML
    - Purpose: Markup language for creating web pages.
    - Usage: Structuring the web pages, including forms, buttons, and other UI components.
10. CSS
    - Purpose: Styling web pages
    - Usage: Applying styles to HTML elements, ensuring a consistent look across the application.
11. JavaScript
    - Purpose: Client-side scripting language for dynamic content.
    - Usage: Handling user interactions, making asynchronous requests to the backend and dynamically updating UI.
12. YouTube IFrame API
    - Purpose: Embedding and controlling YouTube videos.
    - Usage: Dynamically embedding YouTube videos on the page and allowing users to play videos.

14. **Visualization**:

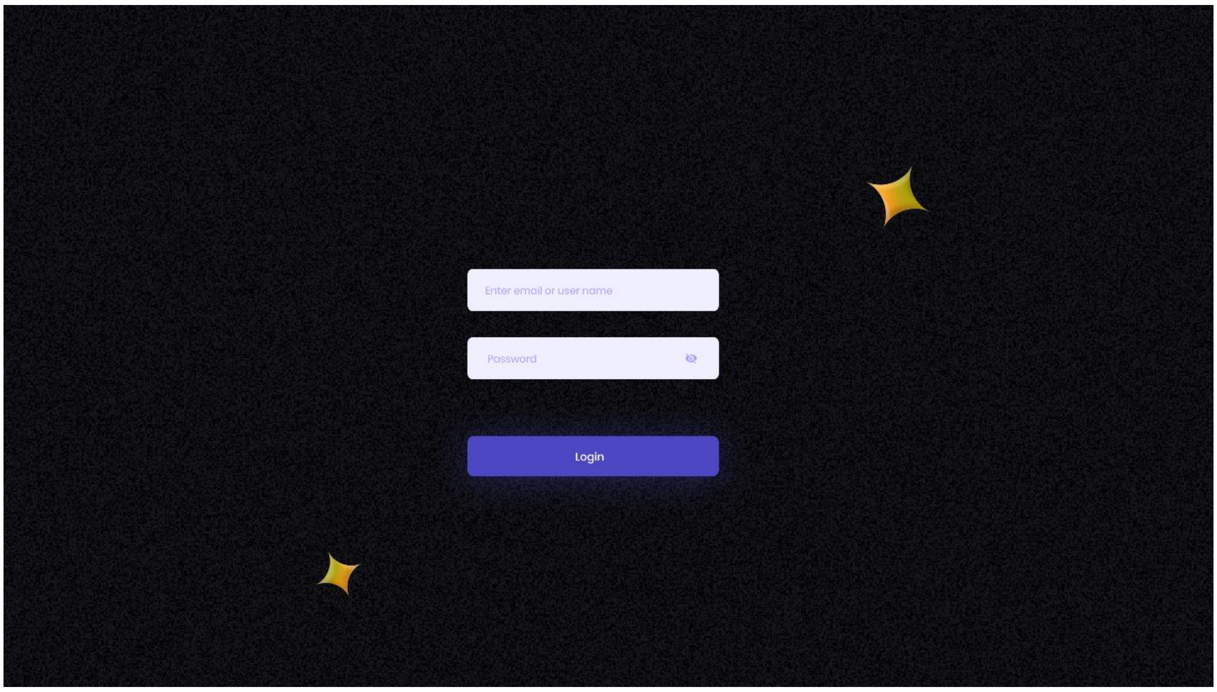
Header page:



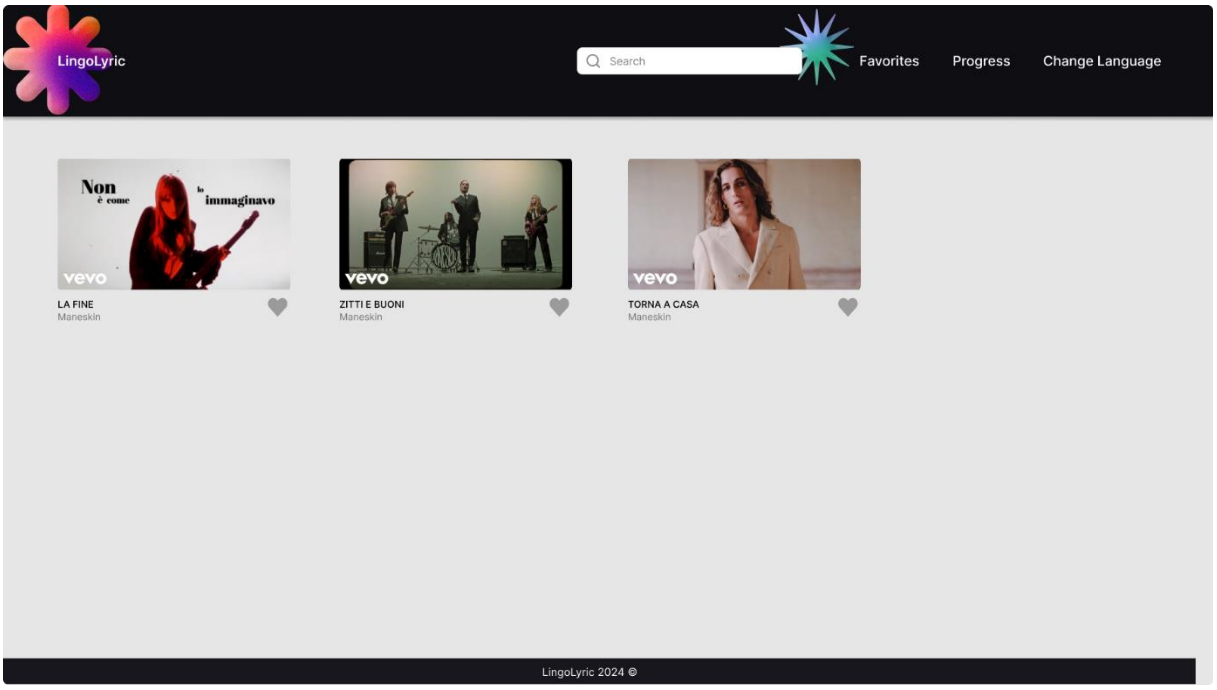
Registration page:



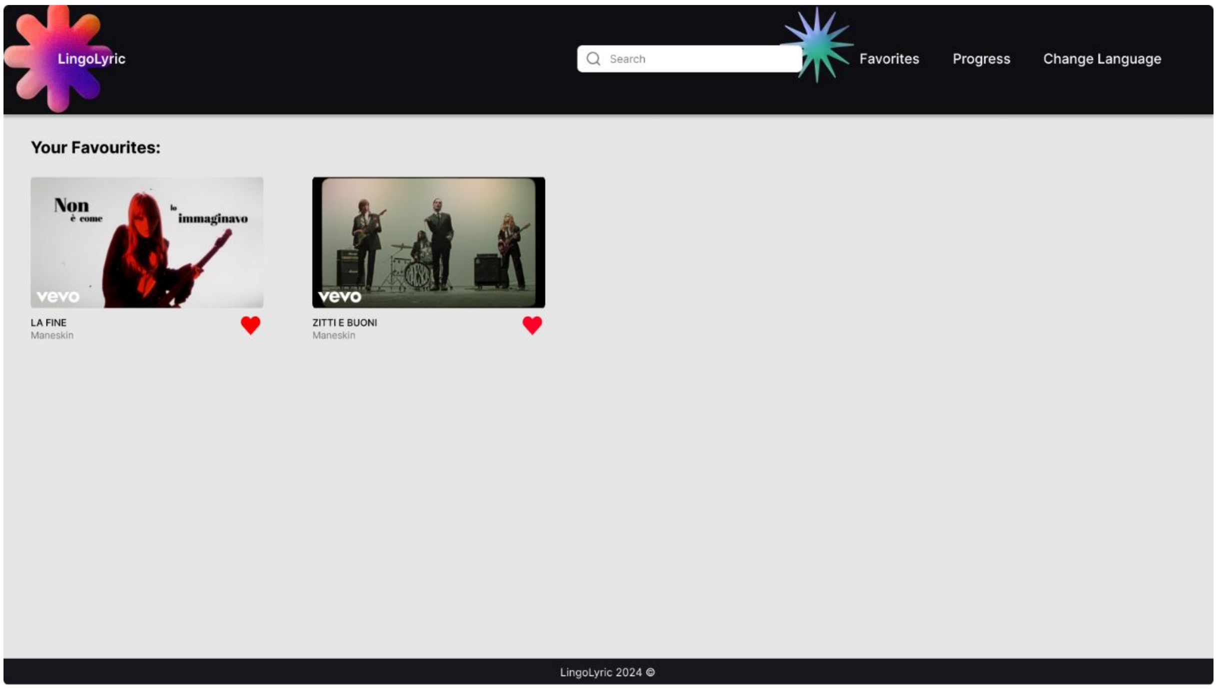
Login page:



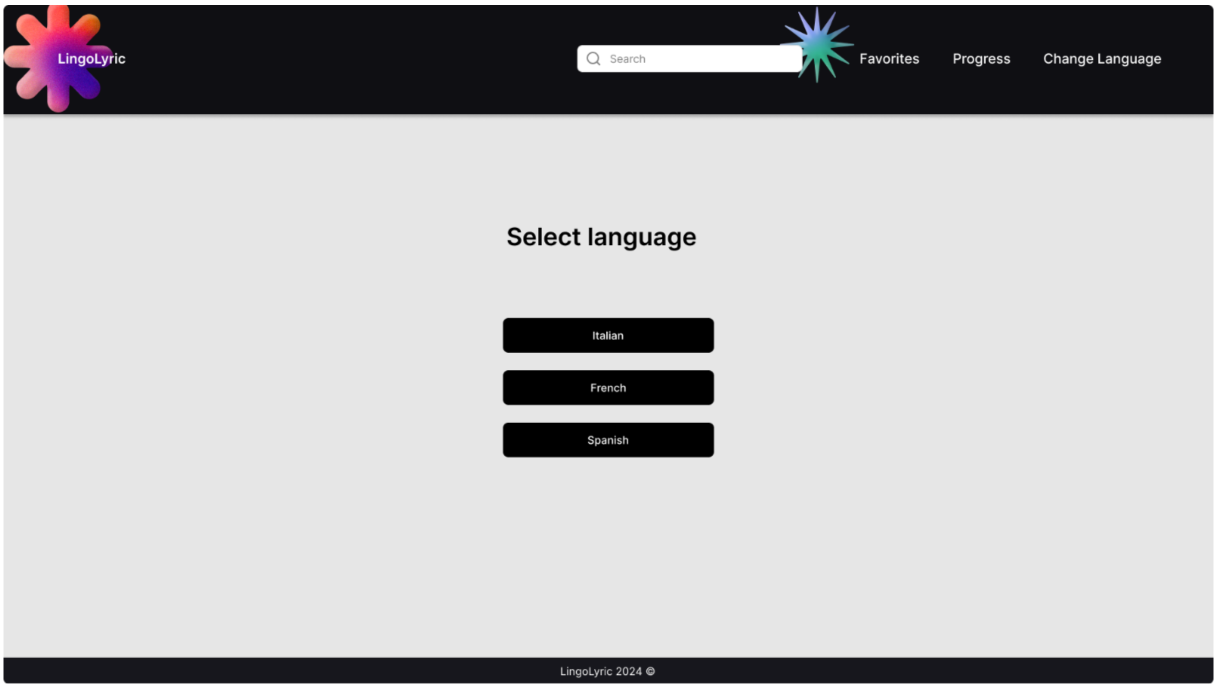
Home page with songs available:



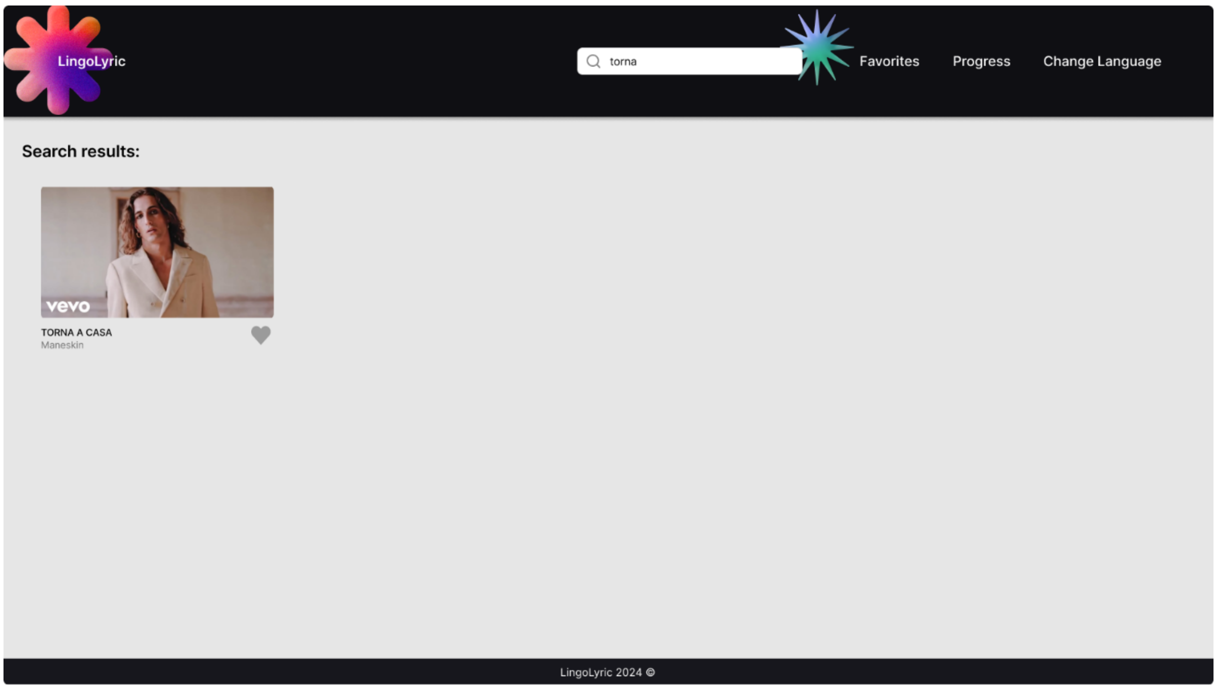
Favourites page:



Language change page:



Search results page:

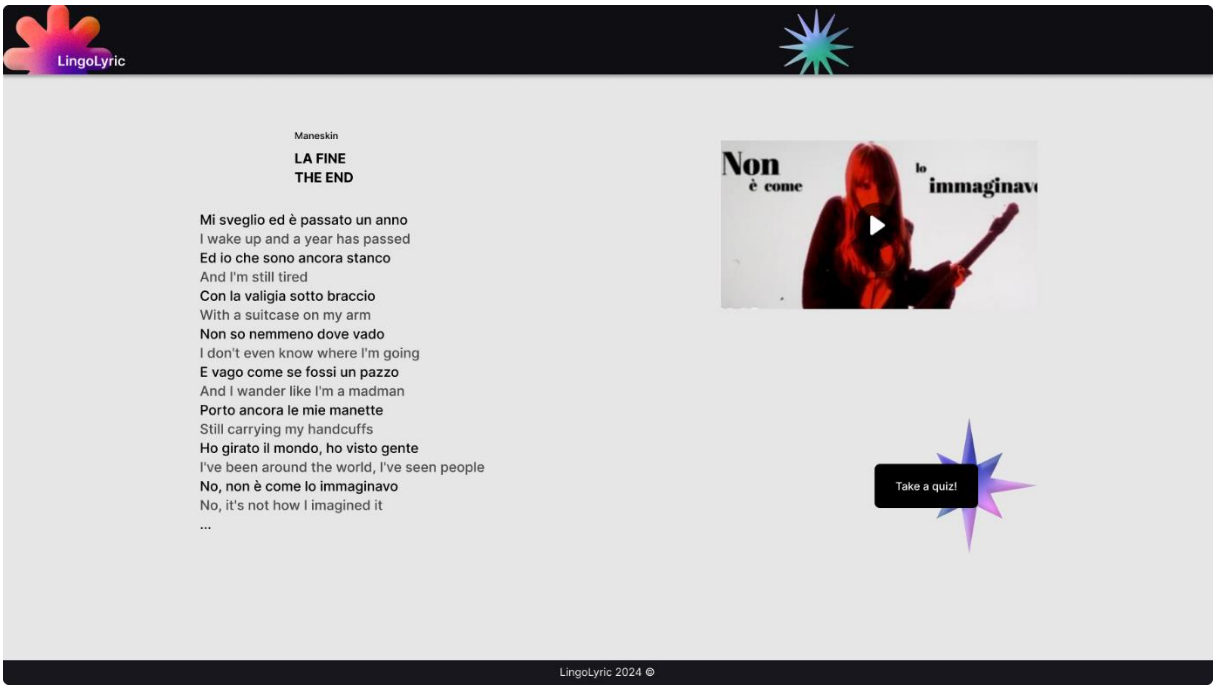


Search results:

TORNA A CASA
Maneskin

LingoLyric 2024 ©

Learn song page:



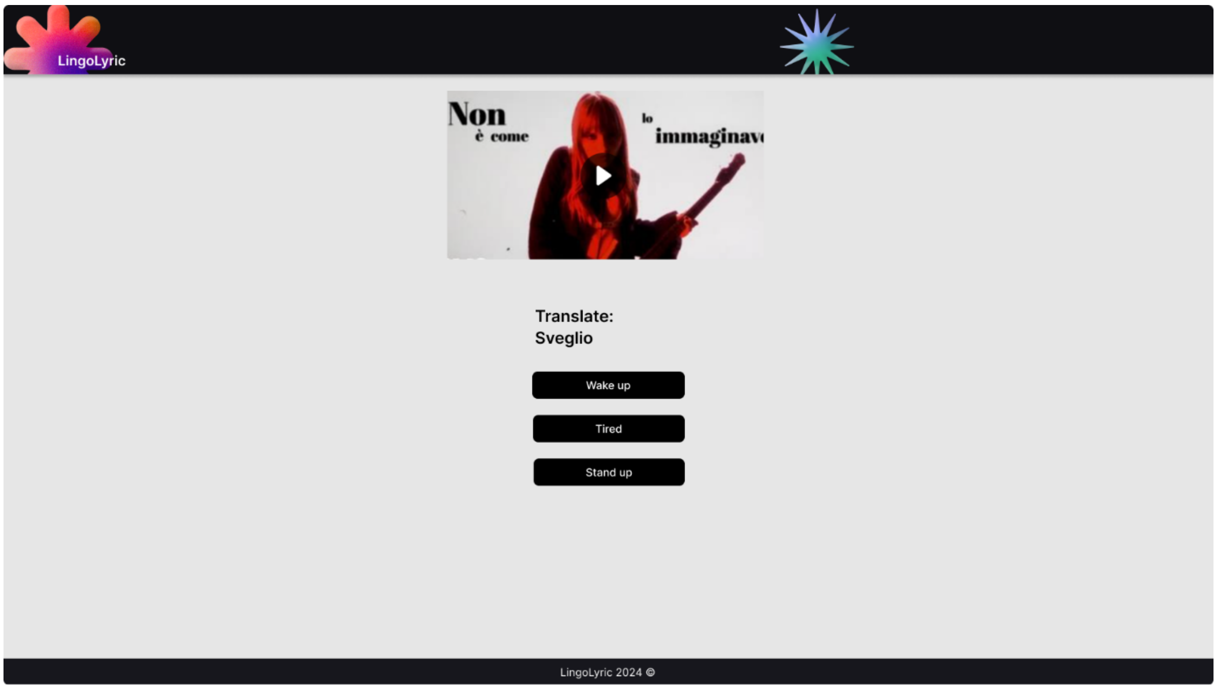
Maneskin

LA FINE
THE END

Mi sveglio ed è passato un anno
I wake up and a year has passed
Ed io che sono ancora stanco
And I'm still tired
Con la valigia sotto braccio
With a suitcase on my arm
Non so nemmeno dove vado
I don't even know where I'm going
E vago come se fossi un pazzo
And I wander like I'm a madman
Porto ancora le mie manette
Still carrying my handcuffs
Ho girato il mondo, ho visto gente
I've been around the world, I've seen people
No, non è come lo immaginavo
No, it's not how I imagined it
...

LingoLyric 2024 ©

Quiz question page:


LingoLyric

Translate:
Sveglio

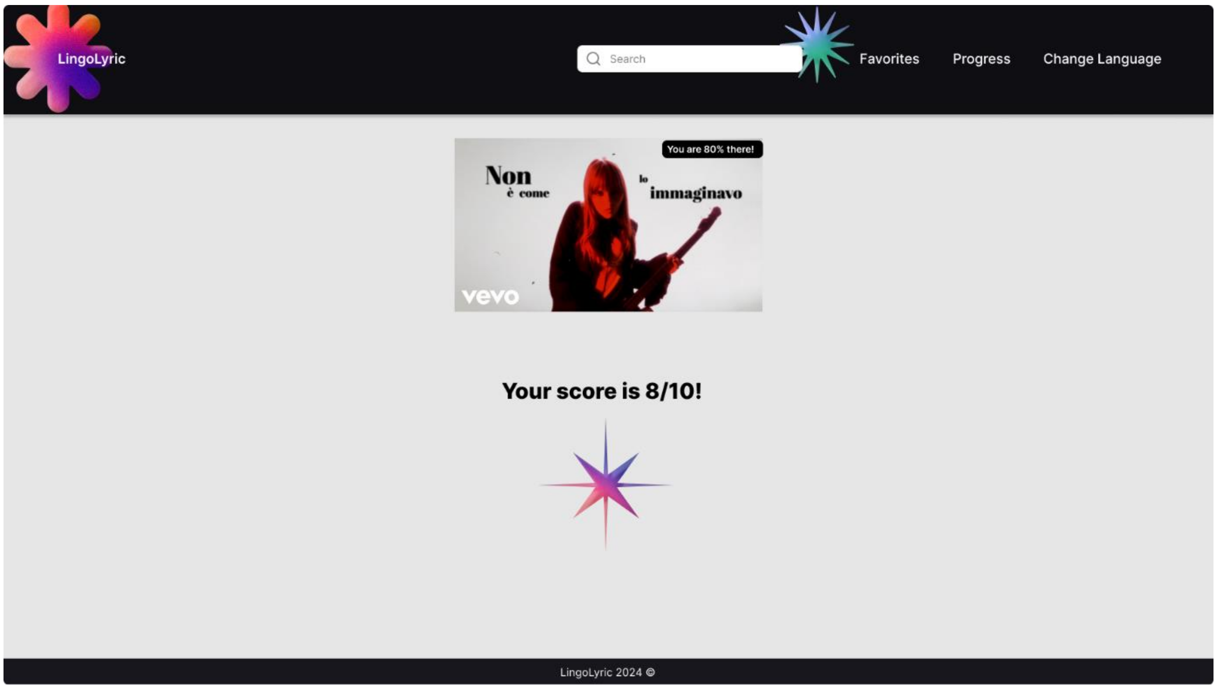
Wake up

Tired

Stand up

LingoLyric 2024 ©

Quiz result page:


LingoLyric

Search

Favorites

Progress

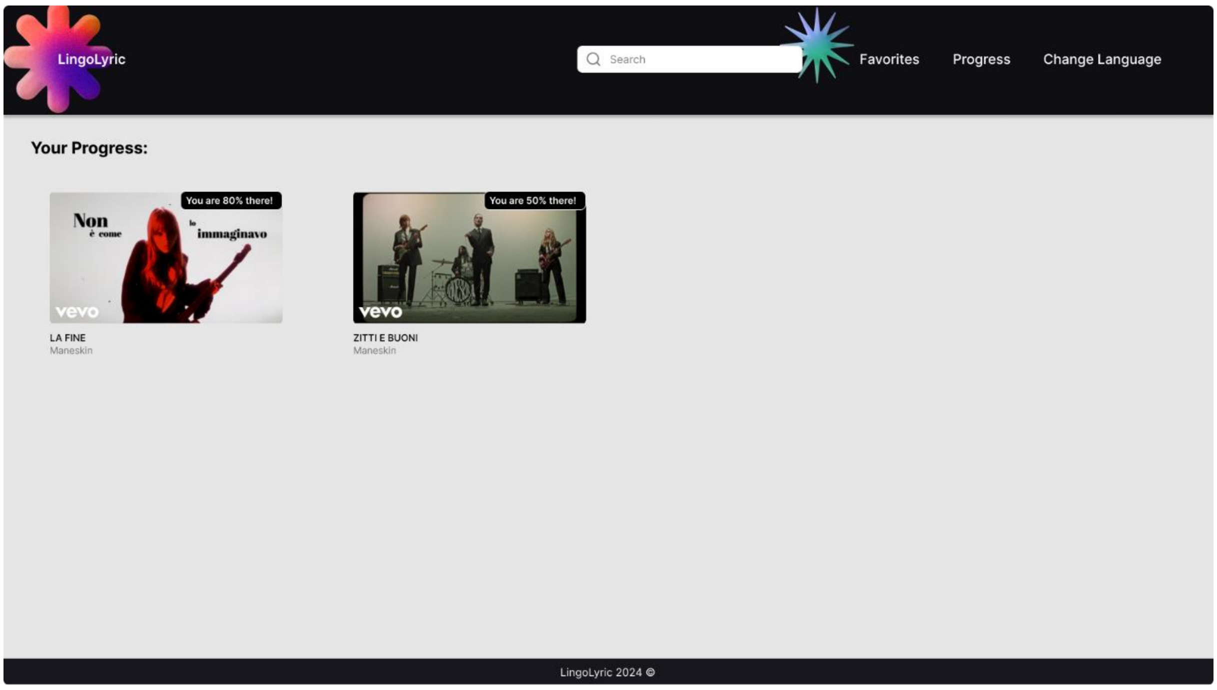
Change Language

You are 80% there!

Your score is 8/10!

LingoLyric 2024 ©

Progress page:



Your Progress:

You are 80% there!
LA FINE
Maneskin

You are 50% there!
ZITTI E BUONI
Maneskin

LingoLyric 2024 ©