

**FULL STACK ASSIGNMENT**

**REPORT ON :**

**SONG RECOMMENDATION SYSTEM**



**SUBMITTED BY : SEC: B**

Neeraj N 22BTAI182 (GITHUB: neerajn142004)

Bharath A 22BTAI191

Lakshmi Narasimha 22BTAI196

Yakub Khan 23BTDS205

**CONTENTS**

* **Introduction**
* **Project Overview**
* **Technology Stack**
* **System Architecture**
* **Implementation**
* **Conclusion**
* **References**

**INTRODUCTION**



**Figure 1.1 Music Chatbot**

**Background:**

* Music plays a crucial role in our lives, providing entertainment, emotional release, and cultural connection.
* With the ever-growing volume of music available online, discovering new songs and artists can be overwhelming.
* Traditional music recommendation systems often rely on complex algorithms and user interactions, which can be less intuitive for some users.

**Objectives:**

* Be Specific: Define measurable goals. For example:
* Achieve at least 80% user satisfaction with recommended songs.
* Implement a chatbot that can handle a minimum of 100 unique user requests per hour.
* Develop a system that accurately identifies user preferences with 90% accuracy.

**Scope:**

* Focus on recommending songs based on user preferences (e.g., artist, genre, mood, decade).
* Incorporate basic conversational abilities, such as greeting, farewell, and handling user inquiries.
* Explore potential integration with popular music streaming platforms (e.g., Spotify, Apple Music).

**Project Overview**

**Description:**

* Provide a detailed narrative: Explain the core functionality of the chatbot.
* Emphasize: How it differs from traditional recommendation systems (e.g., conversational approach, focus on user intent).

**Key Features and Functionalities:**

* Personalized Recommendations: Explain how the chatbot adapts to individual user preferences (e.g., learning from past interactions, using user profiles).
* Conversational Interface: Discuss the implementation of natural language understanding (NLU) techniques (e.g., intent recognition, entity extraction).
* User Preference Handling: Detail how the chatbot captures and processes various user preferences (e.g., artist names, genre keywords, mood descriptions).
* Song Information Retrieval: Explain how the system accesses and presents song information (e.g., artist, album, lyrics, streaming links).
* Chatbot Capabilities: Describe the implementation of basic conversational flows (e.g., greetings, farewells, handling user inquiries, error handling).

**Target Audience:**

* Music enthusiasts seeking to discover new music.
* Users who find traditional music recommendation systems complex or overwhelming.
* Individuals who prefer a conversational and interactive experience.

**Technology Stack**

**Front-End:**

* Option 1: Web-based interface using HTML, CSS, Python and JavaScript (e.g., with a framework like React or Vue.js)
* Option 2: Chatbot platform (e.g., Dialogflow, Rasa) with a user-friendly interface

**Back-End:**

* Programming language: Python
* Framework: Flask (for web-based interface) or custom server implementation
* NLP library: NLTK or spaCy

**Database:**

* SQLite or a cloud-based database (e.g., PostgreSQL, MongoDB) for storing user preferences and song information (if not using an external API)

**APIs:**

* Music streaming platform APIs (e.g., Spotify Web API, Apple Music API) for song data, recommendations, and playback integration.

**System Architecture:**

**High-level architecture diagram,**

**Data flow and interaction between components**

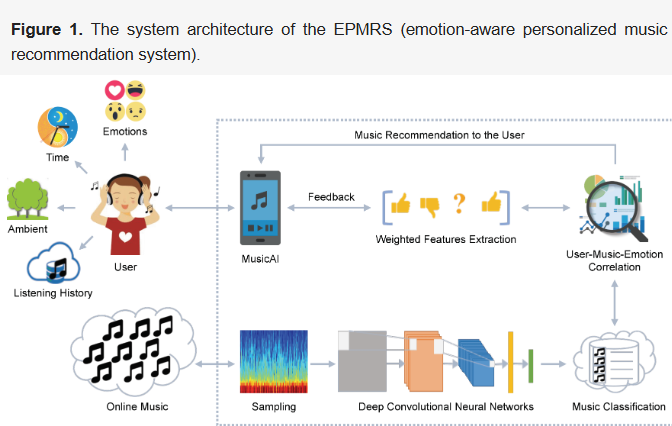


Figure 2:

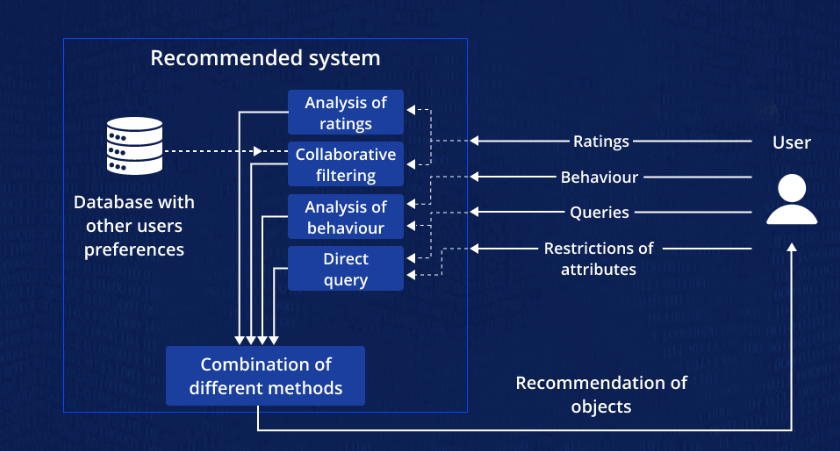


Figure 2: In the Recommendation system chatbot

Figure 3:

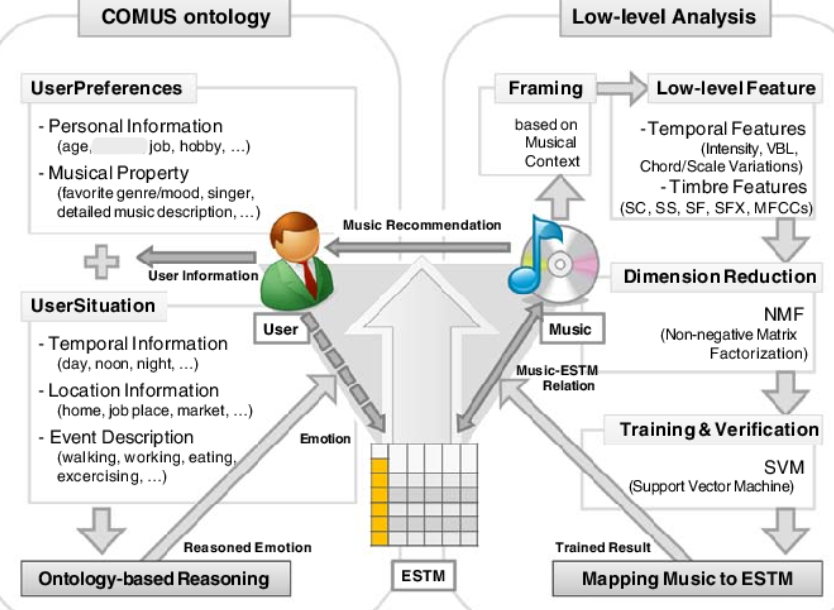
****

Figure 3: Training the recommendation syatem

**Implementation Plan**

Phase 1: Requirements Gathering and Analysis

Task: Conduct user interviews, surveys, and analyze user feedback

Phase 2: Development

Task: Develop the frontend and backend components.

Responsibility: Frontend and Backend Developers.

**Testing and Quality Assurance Procedures**

* Unit Testing: Test individual components in isolation.
* Integration Testing: Test the interaction between different components.
* Performance Testing: Test the system's performance under load.

**Code snippets:**

**Code : index.html**

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8">

    <title>Song Recommendation Chatbot - Rero</title>

    <link rel="stylesheet" href=https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css>

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous"/>

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>

    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN" crossorigin="anonymous"></script>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV" crossorigin="anonymous"></script>

    <link rel="stylesheet" type="text/css" href="kar1.css">

</head>

<body>

    <div class="first-sec d-flex flex-row justify-content-start">

        <h1 class="primary-title"><i class="fas fa-compact-disc mydisk"></i><a href="index.html" class="btn mybtn1">Song Recommendation Chatbot - Rero</a></h1>

        <div class="secondary-titles">

            <a class="top-links btn" href="">About Me</a>

            <a class="top-links btn" href="">Developer info</a>

            <a class="top-links btn" href="">Contact us</a>

        </div>

    </div>

        <div class="main-container d-flex flex-column justify-content-center">

            <div>

                <h1 class="main-title">Any mood, Any time</h1>

                <p>Right music for every mood   </p>

                <a href="main.html" class="mybtn btn">Get Started</a>

            </div>

        </div>

</body>

</html>

**Code: kar1.css**

@import url("https://fonts.googleapis.com/css2?family=Bree+Serif&family=Caveat:wght@400;700&family=Lobster&family=Monoton&family=Open+Sans:ital,wght@0,400;0,700;1,400;1,700&family=Playfair+Display+SC:ital,wght@0,400;0,700;1,700&family=Playfair+Display:ital,wght@0,400;0,700;1,700&family=Roboto:ital,wght@0,400;0,700;1,400;1,700&family=Source+Sans+Pro:ital,wght@0,400;0,700;1,700&family=Work+Sans:ital,wght@0,400;0,700;1,700&display=swap");

body {

    color: #000000;

    font-size: 30px;

    height: 100vh;

    background-image: linear-gradient( 135deg, #FFE985 10%, #FA742B 100%);

    /\*background-color: #F4D03F;\*/

    /\*background-image: linear-gradient(132deg, #F4D03F 0%, #16A085 100%);\*/

    /\*background-color: #FFFFFF;

    background-image: linear-gradient(69deg, #FFFFFF 0%, #6284FF 50%, #FF0000 100%);\*/

    /\*background-image: radial-gradient( circle farthest-corner at -4.5% 34.3%,  rgba(13,20,174,1) 0%, rgba(243,165,140,1) 90% );\*/

    /\*background-image: linear-gradient( 109.6deg,  rgba(255,179,189,1) 1.8%, rgba(254,248,154,1) 50.6%, rgba(161,224,186,1) 100.3% );\*/

    /\*background-image: linear-gradient(180deg, #2af598 0%, #009efd 100%);\*/

    /\*background: #654ea3;   fallback for old browsers \*/

    /\*background: -webkit-linear-gradient(to right, #eaafc8, #654ea3);  /\* Chrome 10-25, Safari 5.1-6 \*/\*/

    /\*background: linear-gradient(to right, #eaafc8, #654ea3);  W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ \*/

}

.first-sec {

    color: #ffffff;

    background-color: rgba(0, 0, 0, 0.5);

    padding: 5px;

    font-family: "Roboto";

}

.primary-title {

    margin-left: 30px;

    padding: 10px;

    /\*font-size: 100px;\*/

}

.secondary-titles {

    /\*text-align: center;\*/

    margin-left: 42%;

    padding-top: 10px;

}

.top-links {

    margin-left: 20px;

    text-decoration: none;

    color: #ffffff;

    font-size: 25px;

    /\*margin-top: 30px;\*/

}

.main-container {

    text-align: center;

    height: 80%;

    /\*background-color: #ffffff;\*/

}

.main-title {

    font-size: 50px;

    font-weight: bold;

}

.mybtn {

    background-color: #47a305;

    color: #ffffff;

    border-radius: 25px;

    width: 150px;

    height: 50px;

    font-size: 20px;

    line-height: 35px;

}

.mybtn1 {

    color: #ffffff;

    font-size: 25px;

}

.mydisk{

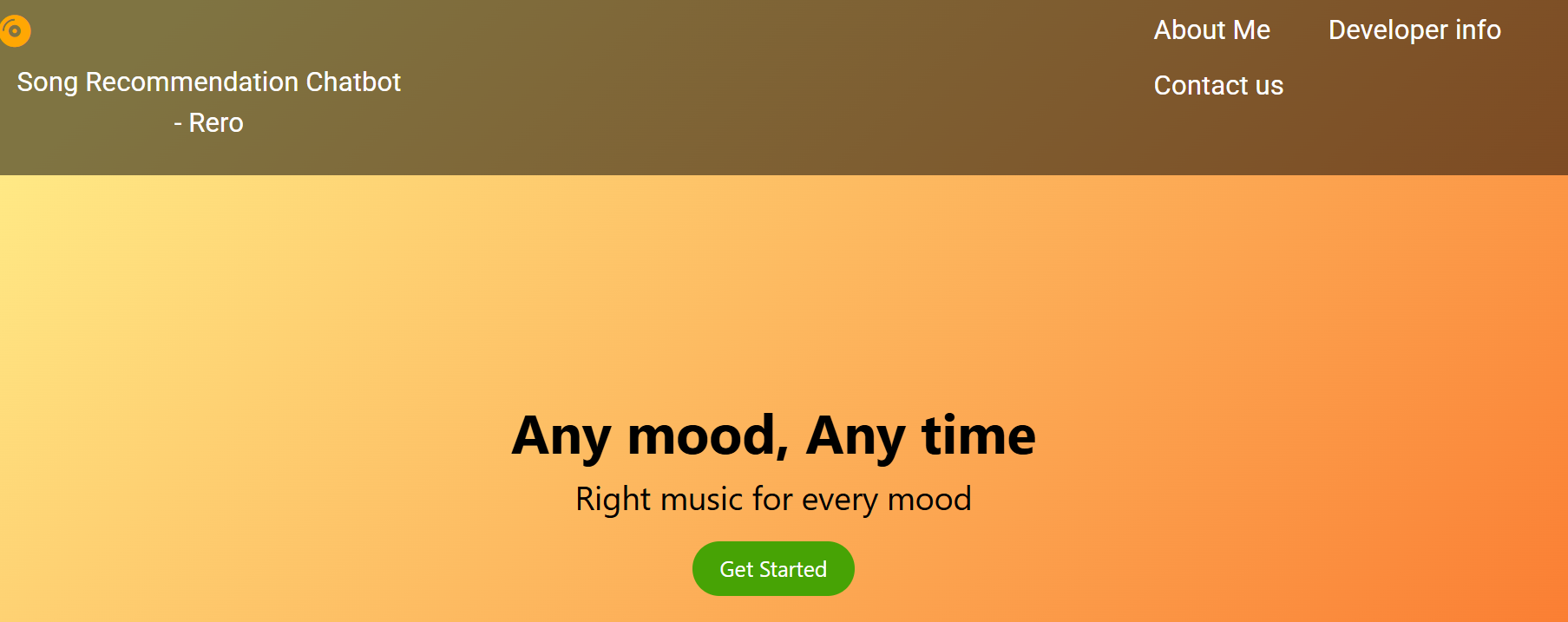
    font-size: 30px;

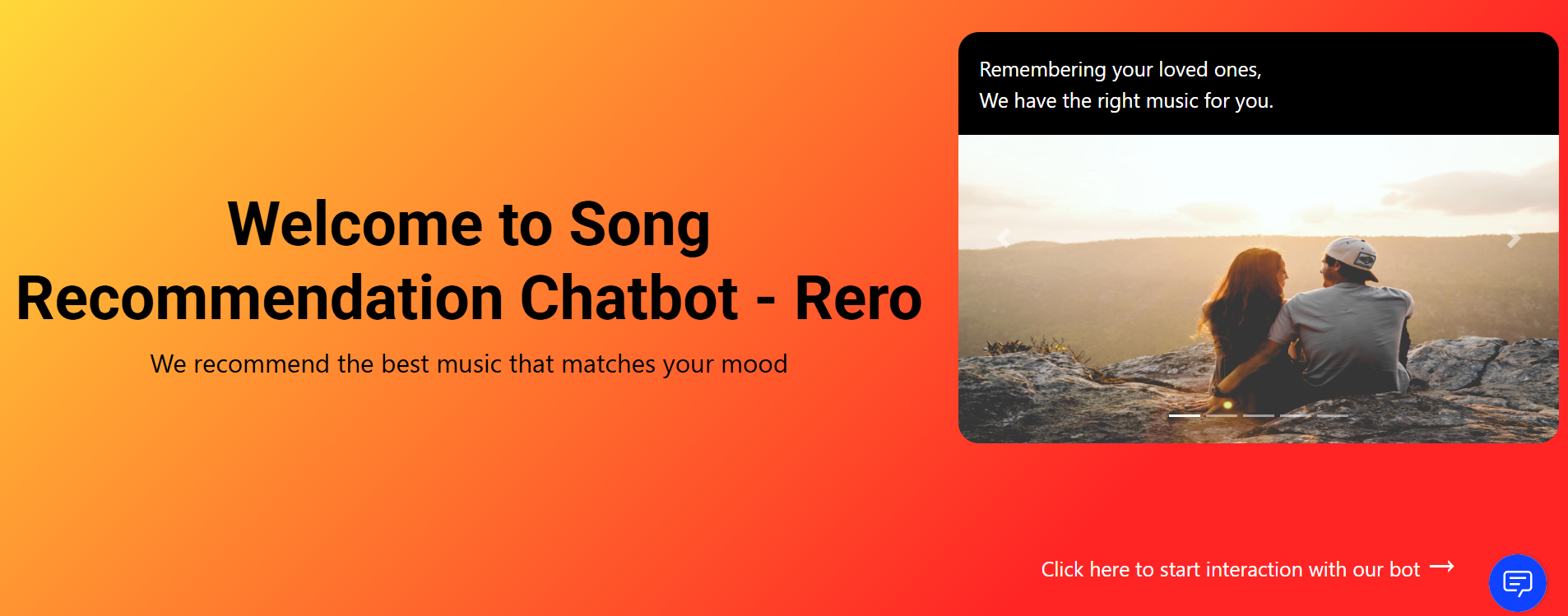
    display: inline-block;

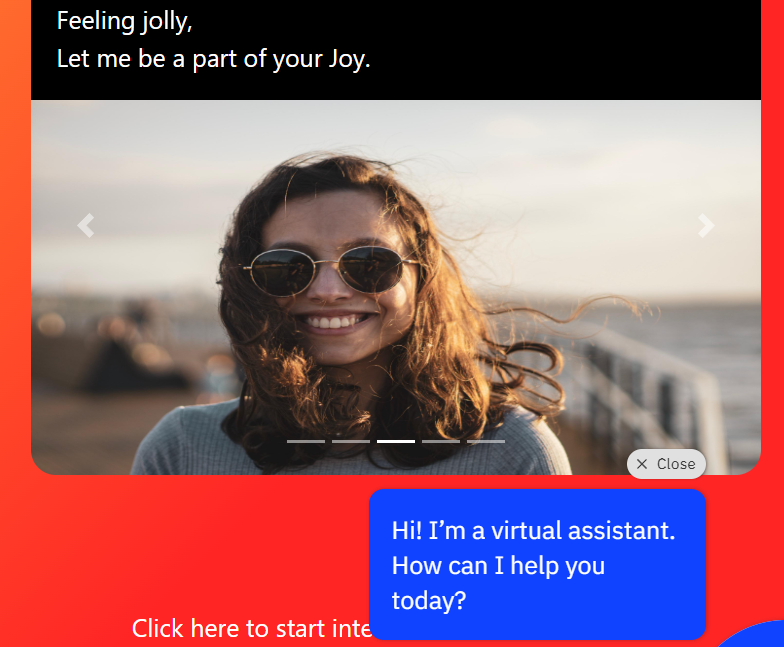
    color: #ffa703;

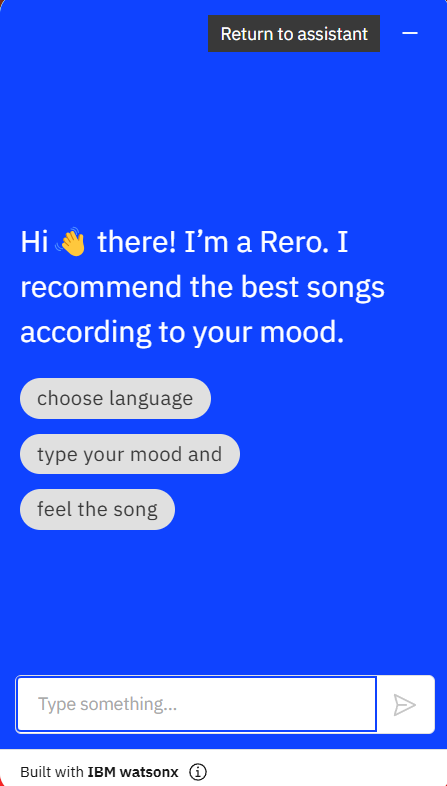
}

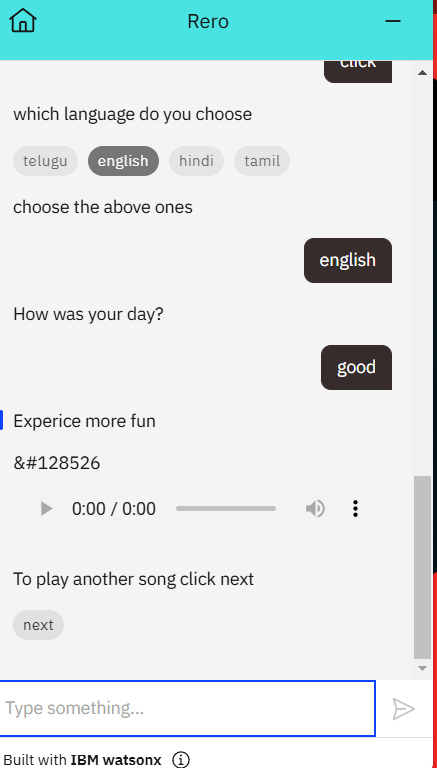
**Screenshots**

****

****

****

****



**Conclusion**

* **Successful Implementation:** The project successfully developed a functional chatbot capable of providing personalized song recommendations based on user preferences.
* **Key Features Achieved:** The chatbot effectively utilizes NLP techniques to understand user input, processes user preferences (artist, genre, mood), and retrieves relevant song recommendations from a music database or API.
* **User-Friendly Interface:** The chatbot offers a user-friendly conversational interface that enhances the music discovery experience.
* **Future Enhancements:**
  + Incorporate advanced emotion recognition capabilities to further personalize recommendations.
  + Expand the system's knowledge base to include a wider range of music genres and artists.
  + Integrate with more music streaming platforms to provide a seamless listening experience.
  + Explore collaborative filtering techniques to refine recommendation accuracy.

**References**

* H.Immanuel James, J. James Anto Arnold, J. Maria Masilla Ruban, M. Tamilarasan and R. Saranya, "Emotion based Music Recommendation System", International Research Journal of Engineering and Technology(IRJET)
* Peter Bae Brandtzæg and Asbjørn Følstad. 2018. Chatbots: Changing User Needs and Motivations. Interactions 25, 5 (2018), 38–43.
* N. Dahlback, A. J ¨ onsson, and L. Ahrenberg. 1993. Wizard of Oz studies why and how. ¨ Knowledge-Based Systems 6, 4 (1993), 258 – 266.
* Steven Dow, Blair MacIntyre, Jaemin Lee, Christopher Oezbek, Jay David Bolter, and Maribeth Gandy. 2005. Wizard of Oz Support Throughout an Iterative Design Process. IEEE Pervasive Computing.