Bilkent University

CS 443 Cloud Computing

Project Final Report: Song Lyrics Translator

Alper Mumcular 21902740

Atakan Onrat 21802520

Ceyda Şahin 21903448

Table of Contents

1. Project Overview

1.1 Project Name and Team

Project Name: Cloud Macchiatos

Team Members:

Alper Mumcular	21902740	CS-4th Year	alper.mumcular@ug.bilkent.edu.tr
Ceyda Şahin	21903448	CS-4th Year	ceyda.sahin@ug.bilkent.edu.tr
Mert Atakan Onrat	21802520	CS-4th Year	atakan.onrat@ug.bilkent.edu.tr

1.2 Purpose

Through the web application we will develop in this project, which is aimed at music lovers and those who are curious about the meaning of the lyrics in their language, they will be able to translate their favorite lyrics into the languages they want. With our application, which will contain approximately 2000 songs, we aim to provide our users with a proper and meaningful translation in the language they want.

1.3 Scope

The scope of the system to be produced is as follows:

- It will be a cloud-native web application and Google Cloud Platform will be used.
- Cloud Translation API will be used to translate the lyrics of a song.
- The system will have a large (5 million song lyrics) database.
- The dataset will be taken from Kaggle [1] and stored in Cloud Storage as a bucket.
- Cloud SQL will be used and MySQL (version 8.0) will be used as an instance type for the required databases.
- Frontend pages will be created using React JS.
- For the backend, Node.js will be used.
- Google App Engine is the technology used to deploy the project.
- Finding the searched song must take less than 1 second.
- Translated lyrics should be as accurate as they can be.

1.4 Out of Scope

The ideas that are out of scope for this project are as follows:

- The project will not include what the lyrics symbolize or what events they refer to.
- The system will not display the meanings of slang words that the Cloud Translation API cannot translate.
- Users cannot search in the application by playing the song. To make it more clear, there are some applications where users can search by playing songs. In this way, they get the necessary information about the song. This kind of search will not be in our project.
- There will be no user login functionality in the project (except for admin purposes).

2. System Architecture

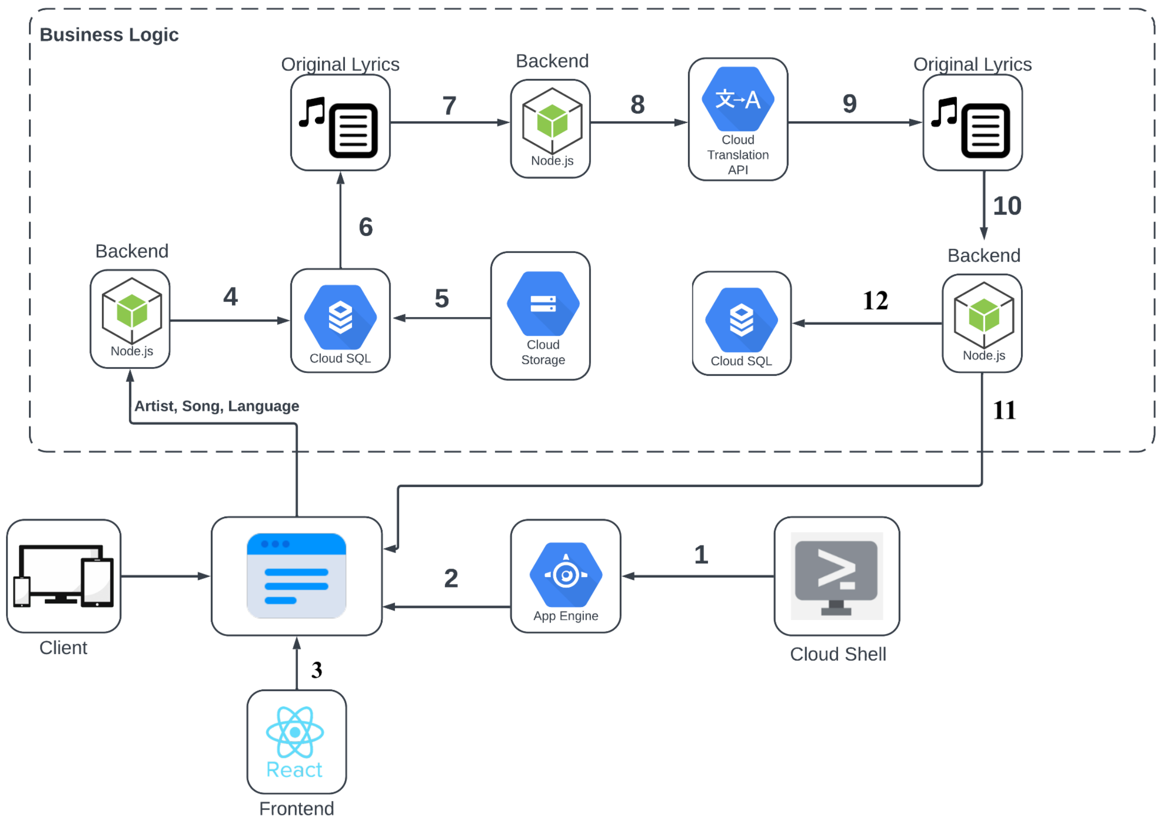


Figure 1: System Diagram of Song Lyrics Translator

1. Write commands to deploy the project using Cloud Shell.
2. Deploy the project using Cloud App Engine.
3. Create a frontend of the project with React.
4. Get results of user input(artist, song) from the database.
5. Create Cloud SQL using Cloud Storage.
6. Copy the results(original lyrics) into a compatible data type.
7. Call Cloud Translation API using backend, which is coded with Node.js
8. Call Cloud Translation API to detect language and translate it to the target language.
9. Copy translated lyrics in a compatible data type.
10. Access and update Cloud SQL using the backend.
11. Use the backend service to repeat the process.
12. Update the database to contain the translated lyrics result.

The system architecture diagram presents an overview of the project's components and their interactions. The application leverages Google Cloud technologies such as Cloud Storage, Cloud SQL, App Engine, and Cloud Translation API. Cloud Functions will be utilized to automate the translation process by interfacing with Cloud Storage and Cloud Translation API to provide the user with the desired output. App Engine will be used to deploy and connect the front end and back end.

The user will search for an artist and song and specify the target language via the frontend server. The input will then be communicated to the backend server through an endpoint. The backend will process the user input and access the Cloud Storage to obtain the original lyrics. If the translated lyrics are present in the database, they will be returned. Otherwise, the original lyrics will be translated using Translation API and copied into a temporary object. The translated lyrics and the target language will then be updated in the database entry for the given artist and song. This process enables the application to provide the user with an accurate and meaningful translation within a short period.

2.1 Choice of Technologies, APIs, Algorithms

2.1.1 Use of a Particular Type of Product

The choice of technologies, APIs, and algorithms for this application was driven by the goal of providing accurate and fast translations to the user. To do this, we decided to use Google Cloud technologies like Cloud Storage, Cloud App Engine, Cloud SQL, and Translation API. The application's backend was developed using Node.js (JavaScript), a popular and effective runtime environment for creating scalable network applications.

React, a widely used and liked JavaScript library for developing user interfaces, was used to build the application's frontend. We selected React because it enables us to build modular, reusable UI components that are simple to manage and update.

We also decided to use the Translation API provided by Google Cloud to perform translations. This API is suitable for our application because it is well-documented and supports a variety of languages.

We used Cloud SQL, a fully managed SQL database service offered by Google Cloud, to manage data storage and persistence. This made it simple for us to store and access data in a scalable and reliable manner.

2.1.2 Future Plans for Extending or Enhancing the Software

In the future, we may improve the accuracy of translations by using machine learning algorithms. We also can integrate speech-to-text and text-to-speech functionality into the application to provide a more comfortable translation experience for the user. Such implementations would also increase the accessibility of our application.

2.1.3 Error Detection and Recovery

In terms of error detection and recovery, we utilized error logging and monitoring tools provided by Google Cloud to identify and address any issues that may arise during runtime.

2.1.4 Communication Mechanisms

Hypertext Transfer Protocol (HTTP) will be used to transmit data between the client and the web server. It will help us to retrieve data from the user, and display the desired output. Since Node.js has a built-in module called HTTP, we are planning to use this module.

2.1.5 Security(Token-based Authentication)

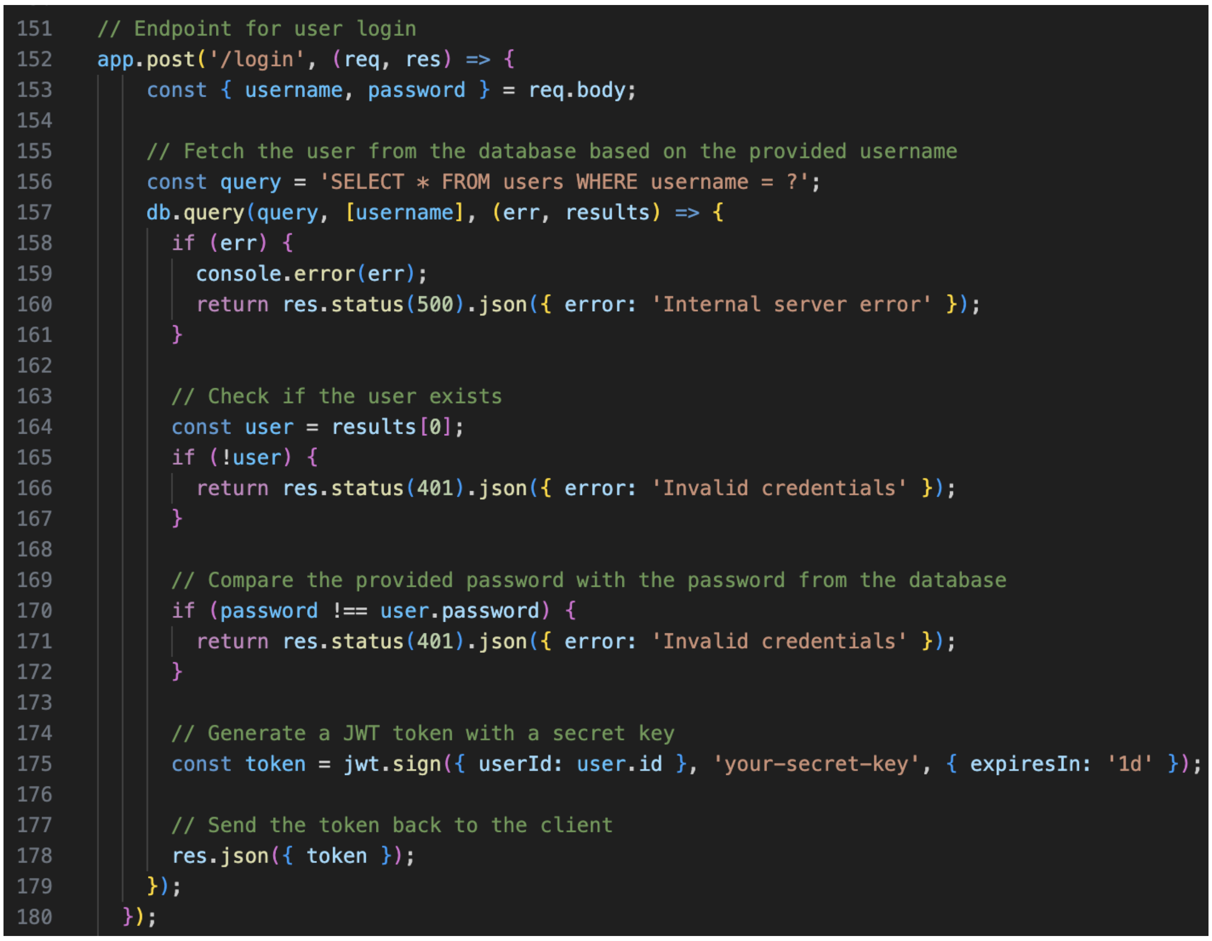
Token-based authentication is used to secure access to the AdminControlPage and ensure that only authenticated users can perform certain actions.

Backend Code:

The backend code handles user login, generates a JWT (JSON Web Token), and includes a middleware function to authenticate requests.

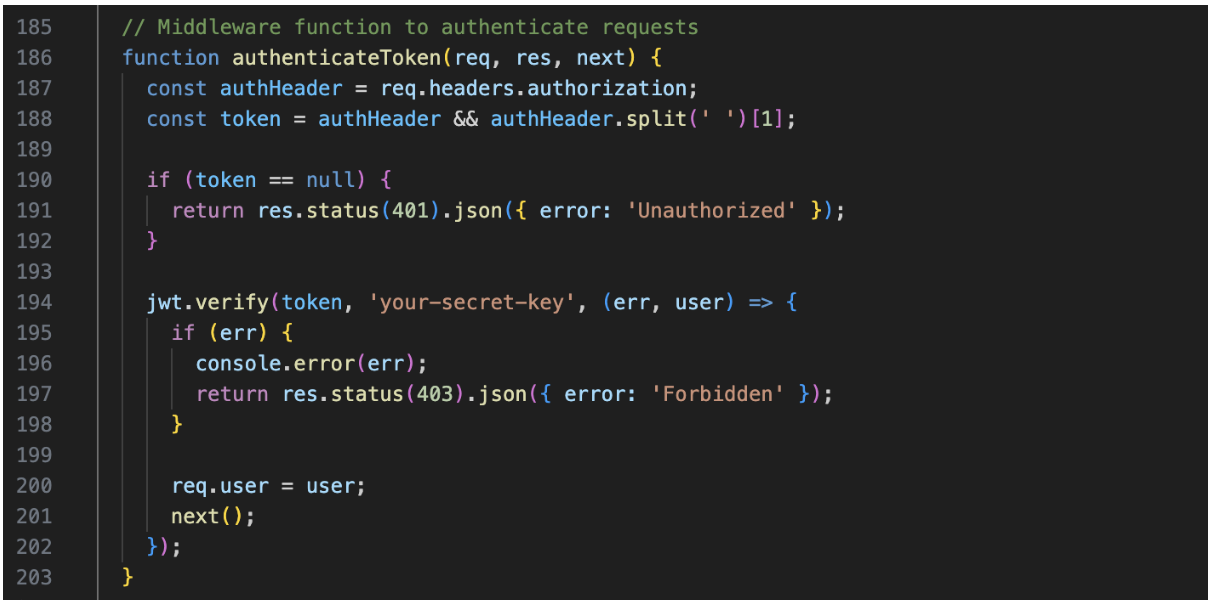
Explanation:

- The /login endpoint receives the username and password from the client.
- The code queries the database to fetch the user based on the provided username.
- If the user exists, the provided password is compared with the password from the database.
- If the credentials are valid, a JWT token is generated using the jsonwebtoken library.
- The token is sent back to the client as a response.
- The authenticateToken middleware function verifies the token sent by the client in the Authorization header.
- If the token is valid, which means the user is authenticated, and the request proceeds to the next middleware or route handler.



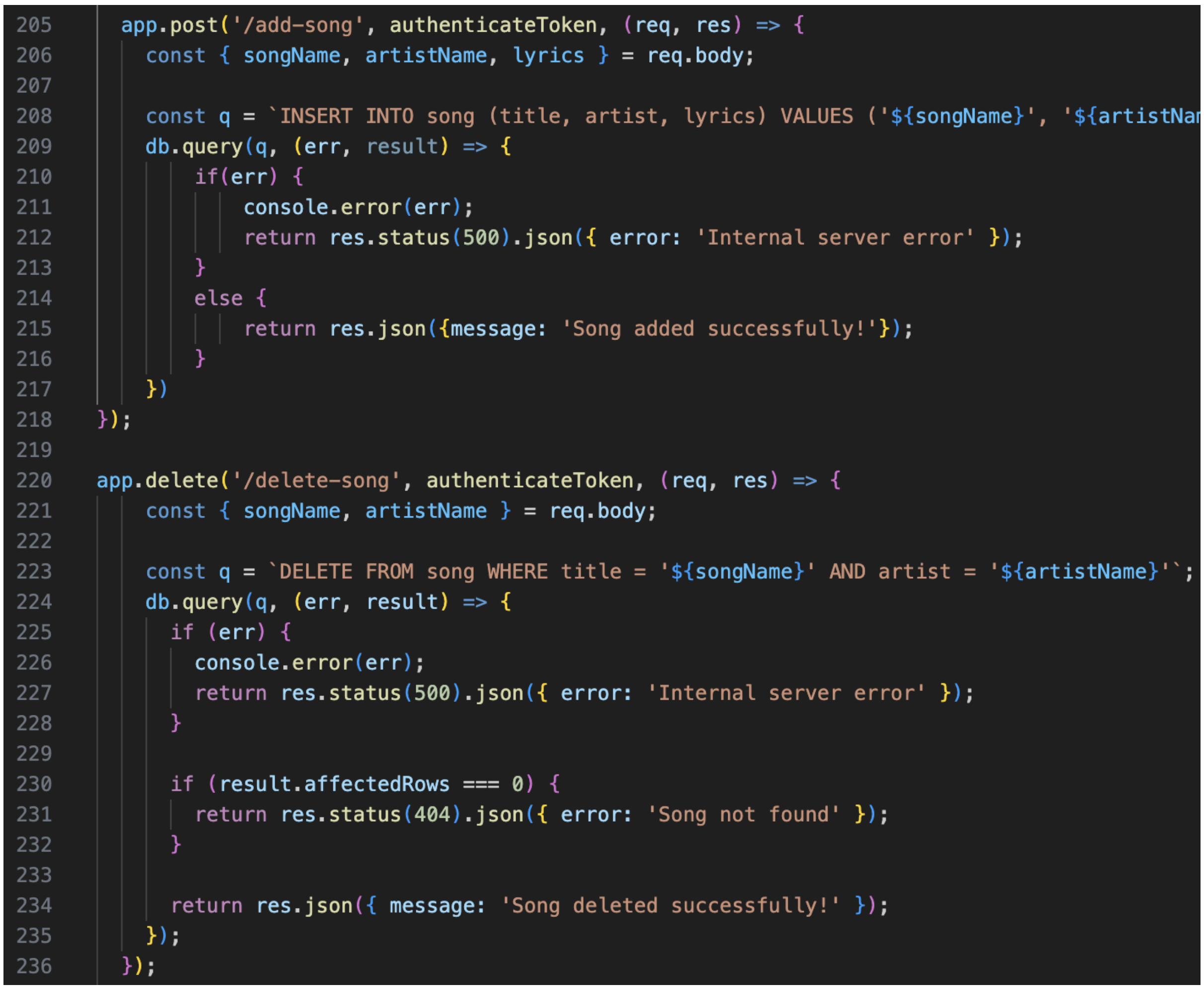
```
// Endpoint for user login
app.post('/login', (req, res) => {
    const { username, password } = req.body;

    // Fetch the user from the database based on the provided username
    const query = 'SELECT * FROM users WHERE username = ?';
    db.query(query, [username], (err, results) => {
      if (err) {
        console.error(err);
        return res.status(500).json({ error: 'Internal server error' });
      }

      // Check if the user exists
      const user = results[0];
      if (!user) {
        return res.status(401).json({ error: 'Invalid credentials' });
      }

      // Compare the provided password with the password from the database
      if (password !== user.password) {
        return res.status(401).json({ error: 'Invalid credentials' });
      }

      // Generate a JWT token with a secret key
      const token = jwt.sign({ userId: user.id }, 'your-secret-key', { expiresIn: '1d' });

      // Send the token back to the client
      res.json({ token });
    });
  });
```

Figure 2: Login endpoint function in backend



```
  // Middleware function to authenticate requests
  function authenticateToken(req, res, next) {
    const authHeader = req.headers.authorization;
    const token = authHeader && authHeader.split(' ')[1];

    if (token == null) {
      return res.status(401).json({ error: 'Unauthorized' });
    }

    jwt.verify(token, 'your-secret-key', (err, user) => {
      if (err) {
        console.error(err);
        return res.status(403).json({ error: 'Forbidden' });
      }

      req.user = user;
      next();
    });
  }
```

Figure 3: authenticateToken function in backend



```
  app.post('/add-song', authenticateToken, (req, res) => {
    const { songName, artistName, lyrics } = req.body;

    const q = `INSERT INTO song (title, artist, lyrics) VALUES ('${songName}', '${artistNa
    db.query(q, (err, result) => {
      if(err) {
        console.error(err);
        return res.status(500).json({ error: 'Internal server error' });
      }
      else {
        return res.json({message: 'Song added successfully!'});
      }
    })
});

app.delete('/delete-song', authenticateToken, (req, res) => {
    const { songName, artistName } = req.body;

    const q = `DELETE FROM song WHERE title = '${songName}' AND artist = '${artistName}'`;
    db.query(q, (err, result) => {
      if (err) {
        console.error(err);
        return res.status(500).json({ error: 'Internal server error' });
      }

      if (result.affectedRows === 0) {
        return res.status(404).json({ error: 'Song not found' });
      }

      return res.json({ message: 'Song deleted successfully!' });
    });
  });
```

Figure 4: add-song and delete-song endpoint functions in backend

Code Snippets:

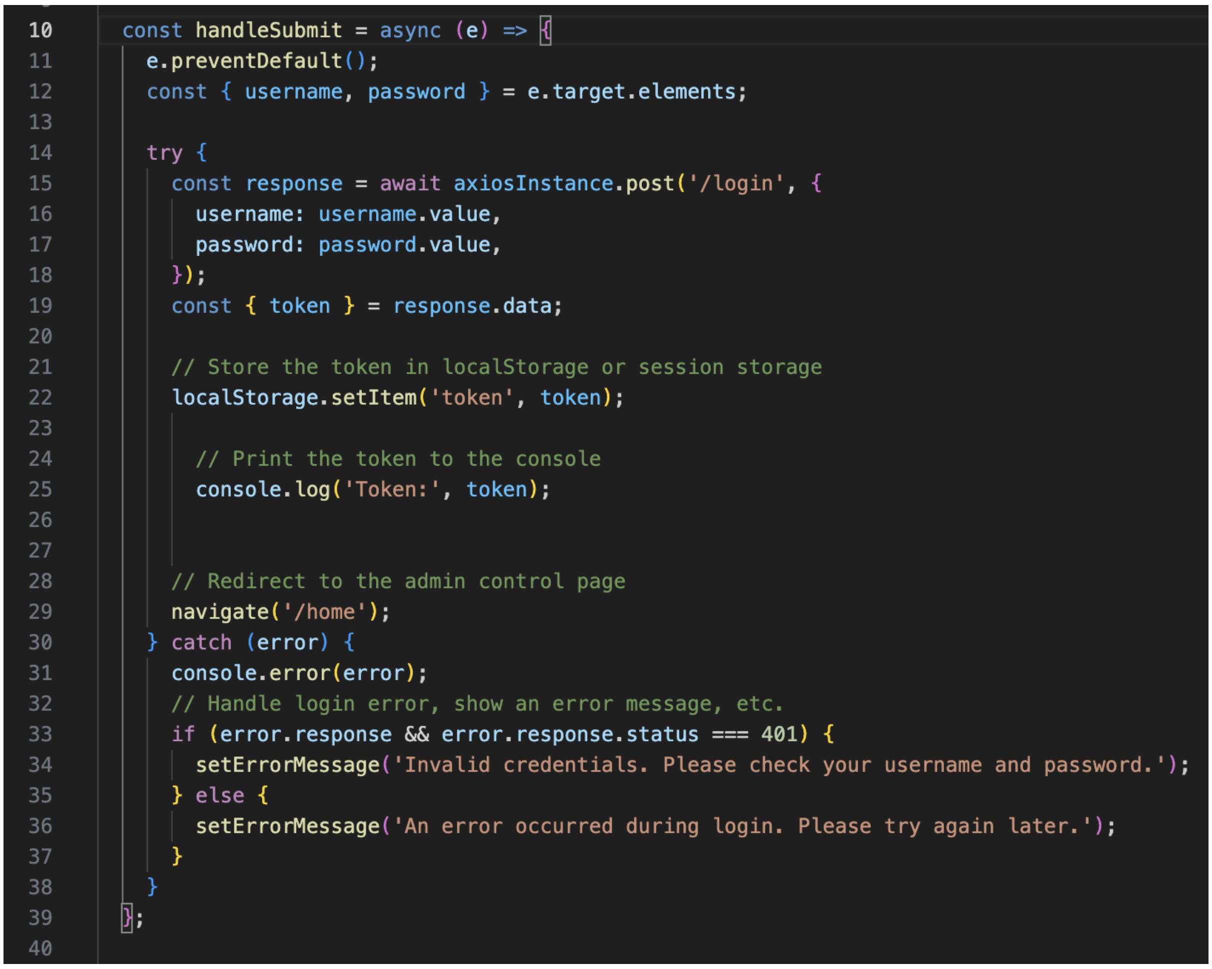
- /add-song and /delete-song routes are protected by the authenticateToken middleware. Only authenticated users with a valid token can access these routes.

Frontend Code:

The frontend code handles the login functionality, stores the token in the browser's localStorage, and redirects the user to the AdminControlPage upon successful login.

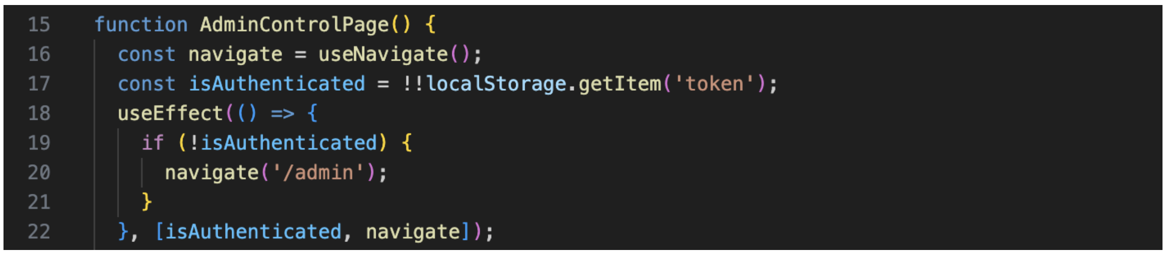
Explanation:

- The handleSubmit function is triggered when the user submits the login form.
- The function sends a POST request to the /login endpoint with the provided username and password.
- If the login is successful, the received token is stored in the browser's Local Storage.
- The user is then redirected to the AdminControlPage.
- The AdminControlPage component checks for the presence of the token in localStorage to determine if the user is authenticated.
- If not authenticated, the user is redirected to the login page(/admin).
- The handleLogout function removes the token from localStorage and redirects the user to the login page.



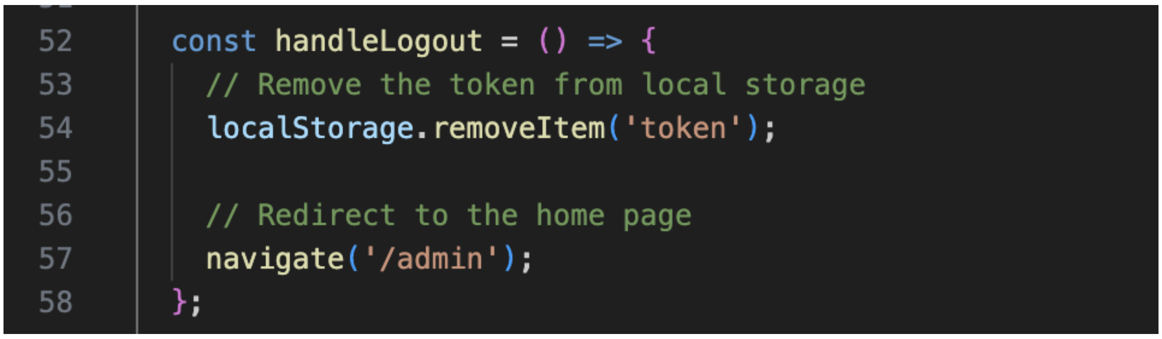
```
const handleSubmit = async (e) => {
  e.preventDefault();
  const { username, password } = e.target.elements;

  try {
    const response = await axiosInstance.post('/login', {
      username: username.value,
      password: password.value,
    });
    const { token } = response.data;

    // Store the token in localStorage or session storage
    localStorage.setItem('token', token);

      // Print the token to the console
      console.log('Token:', token);


    // Redirect to the admin control page
    navigate('/home');
  } catch (error) {
    console.error(error);
    // Handle login error, show an error message, etc.
    if (error.response && error.response.status === 401) {
      setErrorMessage('Invalid credentials. Please check your username and password.');
    } else {
      setErrorMessage('An error occurred during login. Please try again later.');
    }
  }
};
```

Figure 5: Login function in frontend



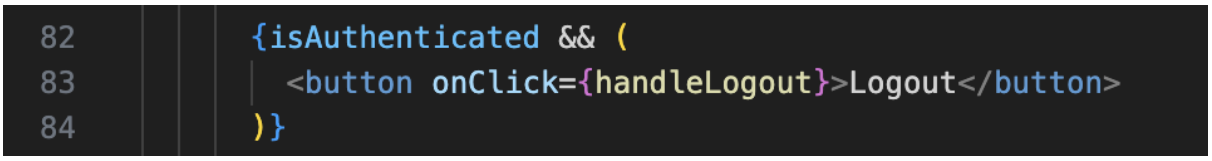
```
function AdminControlPage() {
  const navigate = useNavigate();
  const isAuthenticated = !!localStorage.getItem('token');
  useEffect(() => {
    if (!isAuthenticated) {
      navigate('/admin');
    }
  }, [isAuthenticated, navigate]);
```

Figure 6: Authentication check in frontend



```
const handleLogout = () => {
  // Remove the token from local storage
  localStorage.removeItem('token');

  // Redirect to the home page
  navigate('/admin');
};
```

Figure 7: Logout function



```
{isAuthenticated && (
  <button onClick={handleLogout}>Logout</button>
)}
```

Figure 8: Logout button in frontend

2.1.6 Overall

Overall, the choice of technologies, APIs, and algorithms was based on the goal of providing accurate, secure, and fast translations to the user while ensuring the scalability and reliability of the application.

3. Data Dictionary

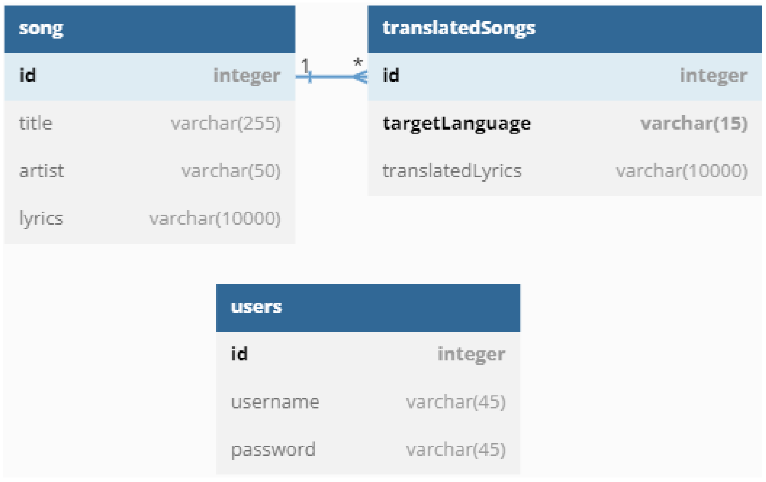


Figure 9: Database Schema Diagram

Description of tables:

Table 1: “song” Table

song

Field	Notes	Type
id	Unique Identifier for songs in the database	INT
title	Title of the song	VARCHAR(255)
artist	Artist of the song	VARCHAR(50
lyrics	Lyrics of the song	VARCHAR(10000)

Table 2: “translatedSongs” Table

translatedSongs

Field	Notes	Type
id	Foreign key to the song table	INT
targetLanguage	Language to be translated	VARCHAR(15)
translatedLyrics	Lyrics in the language to be translated	VARCHAR(10000)

- The “song” table will be used to store songs in their original languages. The “translatedSongs” table will be used to avoid the need to re-translate a previously translated song into the desired language.

Table 3: “users” Table

translatedSongs

Field	Notes	Type
id	Unique identifier for admins in the database	INT
username	Username of the admin	VARCHAR(45)
password	Password of the admin	VARCHAR(45)

- The "users" table was created for authorized people to enter the admin page. It has nothing to do with users who enter the site to translate songs. For admin purposes only.

4. Data Design

4.1 Persistent/Static Data

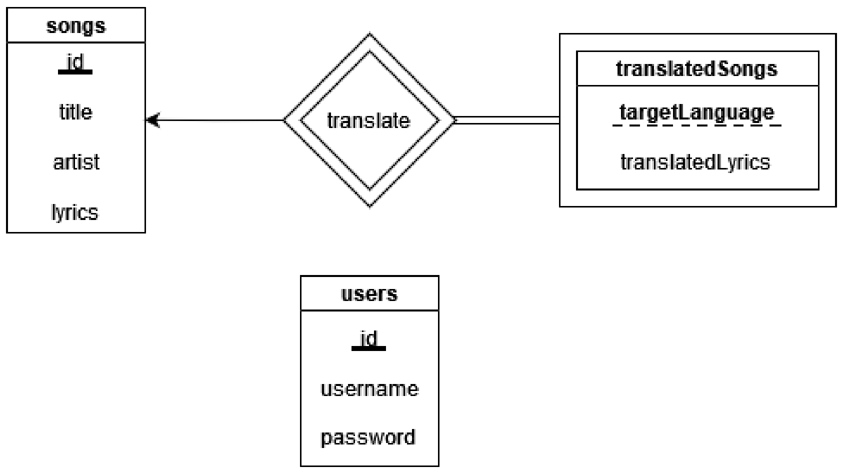


Figure 10: E/R Diagram of Song Lyrics Translator

Relational Schemas:

song(<u>id</u>, title, artist, lyrics, language)

translatedSongs(<u>id, targetLanguage</u>, translatedLyrics)

id: references to song(id)

users(<u>id</u>, username, password)

4.1.1 Static data

The instances added to the ‘song’ and 'users' entities won’t be changed after they are added to them. Therefore these entities can be considered static data. Although admins can change the translated lyrics, the 'translatedSongs' table can also be counted as static data because this operation is not performed very often. Also, project id is also another static data since it is immutable in Google Cloud. Moreover, our CloudSQL instance’s information is also static data. It will be used to manage our database.

4.1.2 Persisted data

The song translator application has a simple data model consisting of three entities. The ‘song’, ‘translatedSongs’ and ‘users’ entities can be considered persisted data since they are kept in the database. The E/R diagram of the database can be seen in Figure 2.

4.2 Transient/Dynamic Data

In order to avoid the cost of Cloud Translation API, we will keep the translated song in the database. Therefore we don’t have transient/dynamic data in our project. However, if we didn't keep the translated songs in the database, these would be our transient data.

4.3 External Interface Data

This section is not applicable (N/A) to our project.

4.4 Transformation of Data

When the user wants to translate a song, we will translate the lyrics of that song, by getting lyrics from the "song" entity, into the desired language using the Cloud Translation API and adding it to the "translatedSongs" entity. In brief, the “translatedLyrics” field will be obtained by translating the original lyrics into the desired language.

5. Detailed of Backend / Cloud Design

5.1 Software Application Domain Chart

By using an application domain model, we represent the business flow by highlighting core components and modules. Figure 4 depicts the domain model for Song Lyrics Translator in UML. The main components are defined as the Front End, the Web Translation Service, Google API and Services, and Data Store.

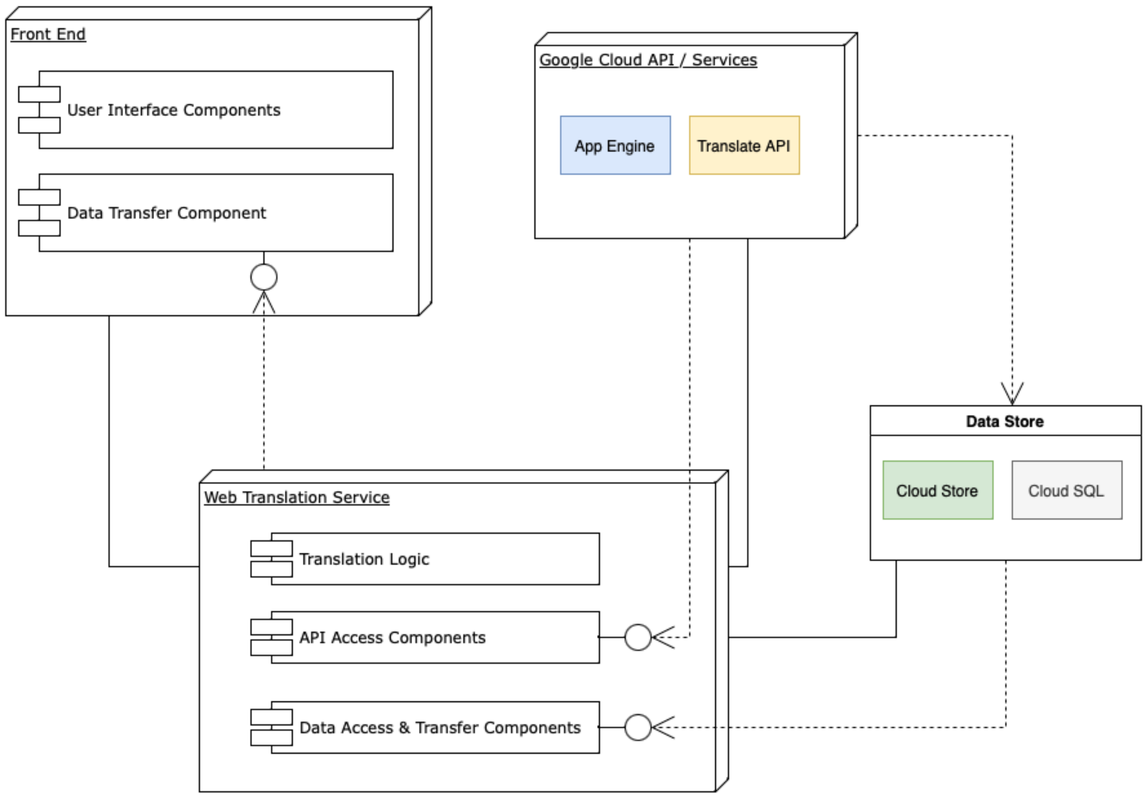


Figure 11 : Application domain model for Song Lyrics Translator

5.2 Components

5.2.1 API Access Components:

The Google Translate API receives a JSON data to pass the data through a translate function. It is structured as follows:

```
{
    "title": "My Heart Will Go On",
    "artist": "Celine Dion",
    "lyrics": "Every night in my dreams\nI see you, I feel you\nThat is how I know you go on\nFar across the distance\nAnd spaces between us\nYou have come to show you go on\nNear, far, wherever you are\nI believe that the heart does go on\nOnce more, you open the door\nAnd you're here in my heart\nAnd my heart will go on and on\nLove can touch us one time\nAnd last for a lifetime\nAnd never let go 'til we're gone\nLove was when I loved you\nOne true time I'd hold to\nIn my life, we'll always go on\nNear, far, wherever you are\nI believe that the heart does go on (why does the heart go on?)\nOnce more, you open the door\nAnd you're here in my heart\nAnd my heart will go on and on\nYou're here, there's nothing I fear\nAnd I know that my heart will go on\nWe'll stay forever this way\nYou are safe in my heart and\nMy heart will go on and on",
    "id": 1
  }
```

Once this input is processed and translated, the API response is structured as follows:

```
{
  "translatedData": "Her gece rüyamda\nseni görüyorum, seni hissediyorum\ndevam ettiğini böyle biliyorum\nUzak mesafe boyunca\nVe aramızdaki boşluklar\ndevam ettiğini göstermek için geldin\nYakın uzak Nerede olursanız olun\nKalbin devam ettiğine inanıyorum\nBir kez daha kapıyı açtığında\nve sen burada kalbimdesin\nVe kalbim devam edecek ve devam edecek\nSevgi bir kez bize dokunabilir\nVe bir ömür boyu sürer\nVe biz gidene kadar asla bırakma...",
  "originalLyrics": "Every night in my dreams\nI see you, I feel you\nThat is how I know you go on\nFar across the distance\nAnd spaces between us\nYou have come to show you go on\nNear, far, wherever you are\nI believe that the heart does go on\nOnce more, you open the door\nAnd you're here in my heart\nAnd my heart will go on and on\nLove can touch us one time\nAnd last for a lifetime\nAnd never let go 'til we're gone…”
}
```

5.2.2 Front End

The front end serves the user interface components that communicate with the backend (web translation service). The user fills in the song search form (see Figure 5) which includes the important parameters: *song artist, song name*, *and target language*. The data transfer component sends the required parameters to the web translation service.

5.2.3 Web Translation Service

The backend serves the necessary components such as the translation logic which processes and refines the data returned from API access components. API access components provide the connection with Google Cloud API and services. The data access and transfer components connect to Cloud Store and Cloud SQL whenever a load or store is needed from the song database. In a simple scenario; the parameters are sent from the front end and the data access components connect to Cloud SQL to obtain the original song lyrics. Via API access components, these lyrics are sent to Translate API in detect language mode (default). If there are multiple languages, they are translated separately. The final result is added to the database via data access and transfer components.

6. User Interface Design

6.1 User Interface Design Overview

Song Lyrics Translator is designed to have a simple user interface with no distractions. The user can enter the song artist and name along with their target language and click ‘Search’ (see Figure 5). If the song exists in our database, the application returns a page with the translated lyrics of the song (see Figure 9).

6.1.1 User Point of View

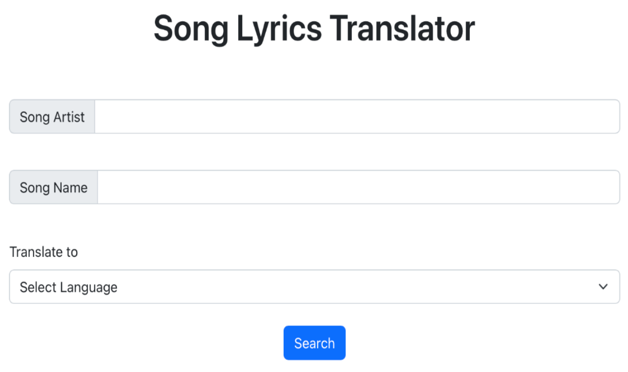
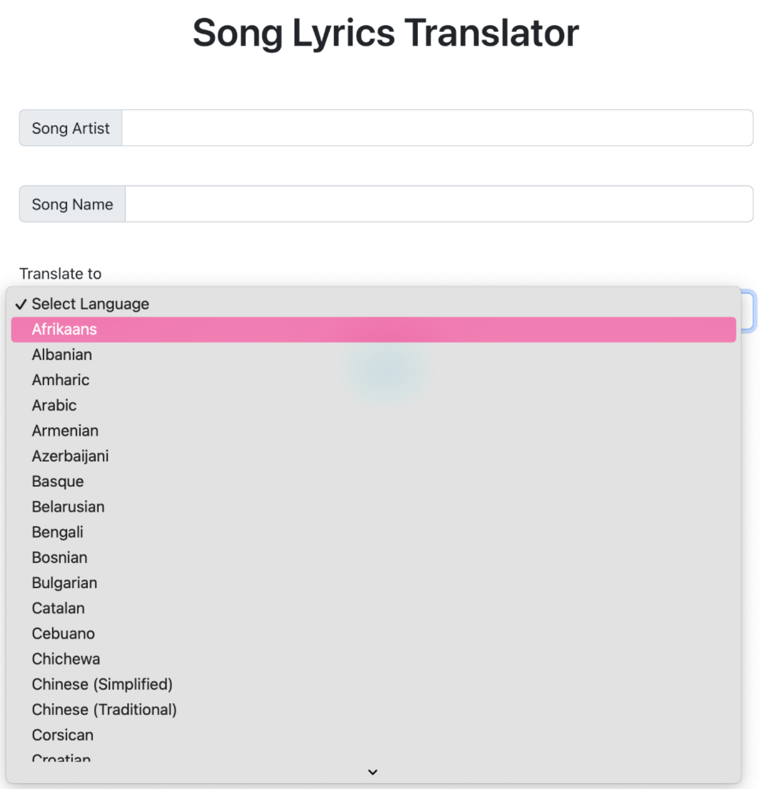


Figure 12: Song Lyrics Translator Main Page


Song Lyrics Translator

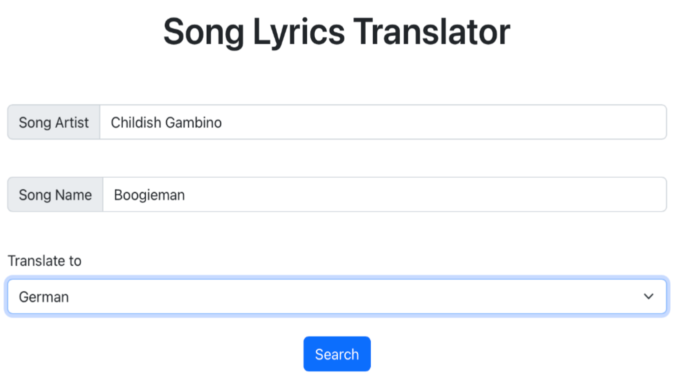
Song Artist

Song Name

Translate to

✓ Select Language
Afrikaans
Albanian
Amharic
Arabic
Armenian
Azerbaijani
Basque
Belarusian
Bengali
Bosnian
Bulgarian
Catalan
Cebuano
Chichewa
Chinese (Simplified)
Chinese (Traditional)
Corsican
Croatian

Figure 13: Song Lyrics Translator Main Page - Select Language


Song Lyrics Translator

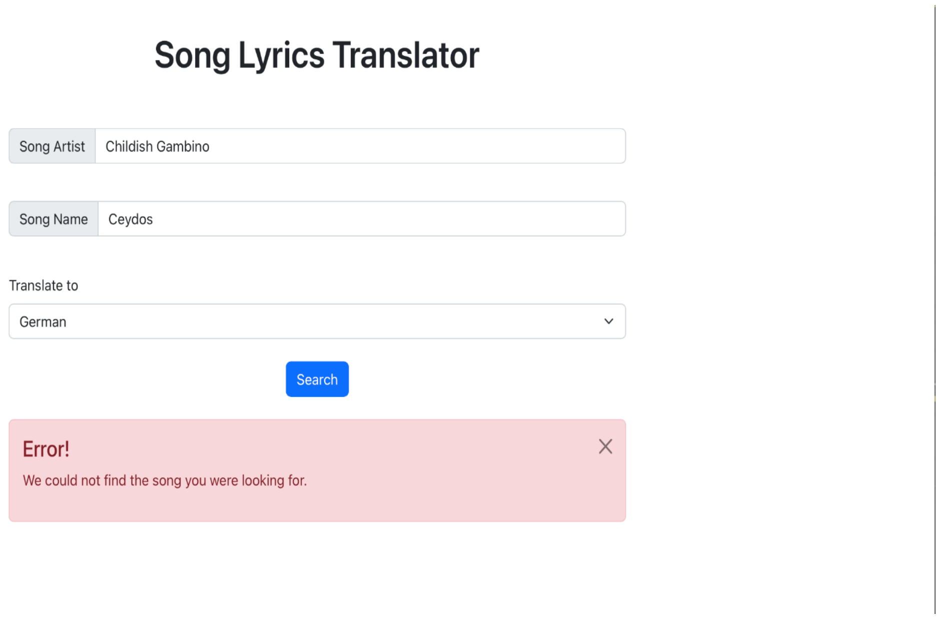
Song Artist: Childish Gambino

Song Name: Boogieman

Translate to: German

Search

Figure 14: Song Lyrics Translator Main Page - User Input


Song Lyrics Translator

Song Artist: Childish Gambino

Song Name: Ceydos

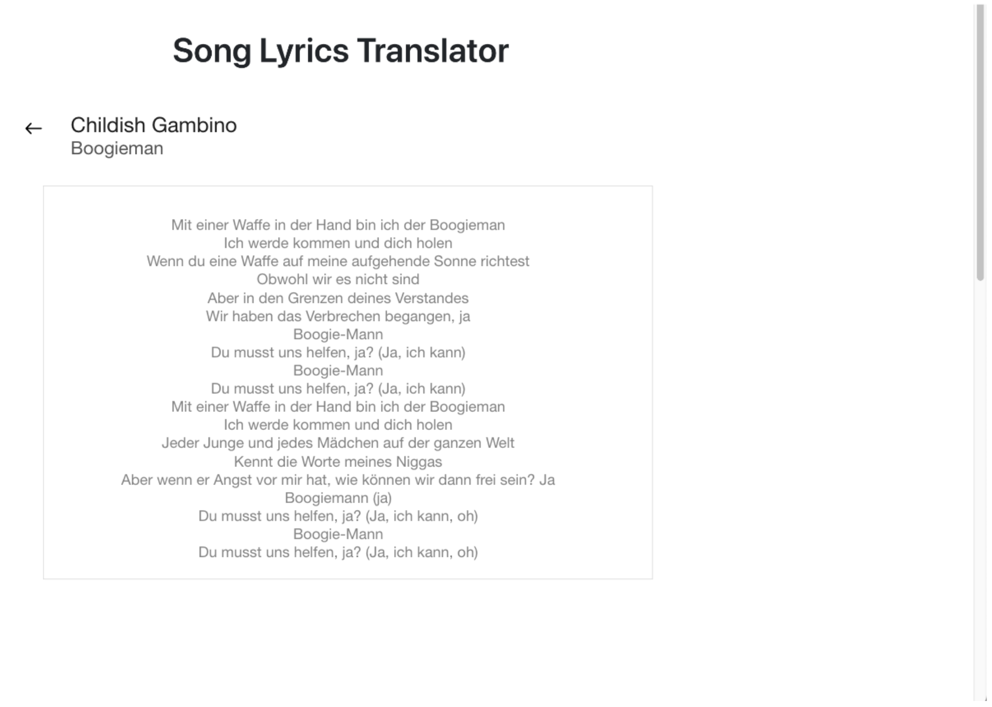
Translate to: German

Search

Error!

We could not find the song you were looking for.

Figure 15: Song Lyrics Translator Main Page - Error Message

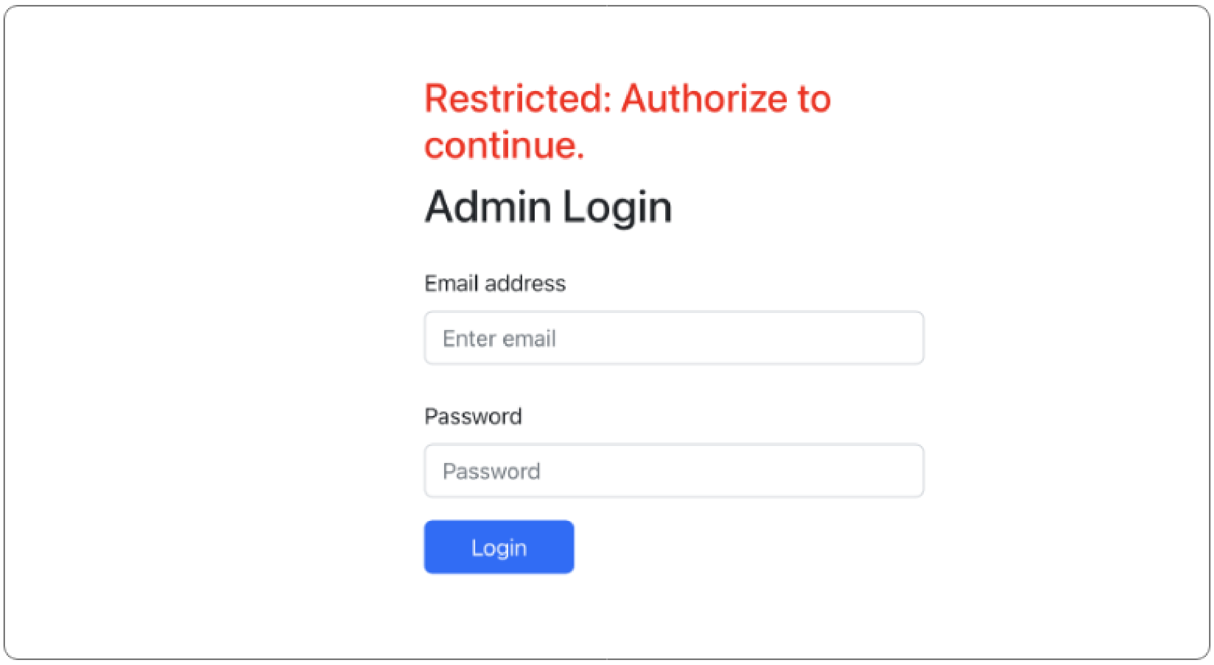

Song Lyrics Translator

Childish Gambino
Boogieman

Mit einer Waffe in der Hand bin ich der Boogieman
Ich werde kommen und dich holen
Wenn du eine Waffe auf meine aufgehende Sonne richtest
Obwohl wir es nicht sind
Aber in den Grenzen deines Verstandes
Wir haben das Verbrechen begangen, ja
Boogie-Mann
Du musst uns helfen, ja? (Ja, ich kann)
Boogie-Mann
Du musst uns helfen, ja? (Ja, ich kann)
Mit einer Waffe in der Hand bin ich der Boogieman
Ich werde kommen und dich holen
Jeder Junge und jedes Mädchen auf der ganzen Welt
Kennt die Worte meines Niggas
Aber wenn er Angst vor mir hat, wie können wir dann frei sein? Ja
Boogiemann (ja)
Du musst uns helfen, ja? (Ja, ich kann, oh)
Boogie-Mann
Du musst uns helfen, ja? (Ja, ich kann, oh)

Figure 16: Song Lyrics Translator Translated Lyrics Page

6.1.2 Admin Point of View


Restricted: Authorize to continue.

Admin Login

Email address

Enter email

Password

Password

Figure 17: Admin Page (unauthorized)

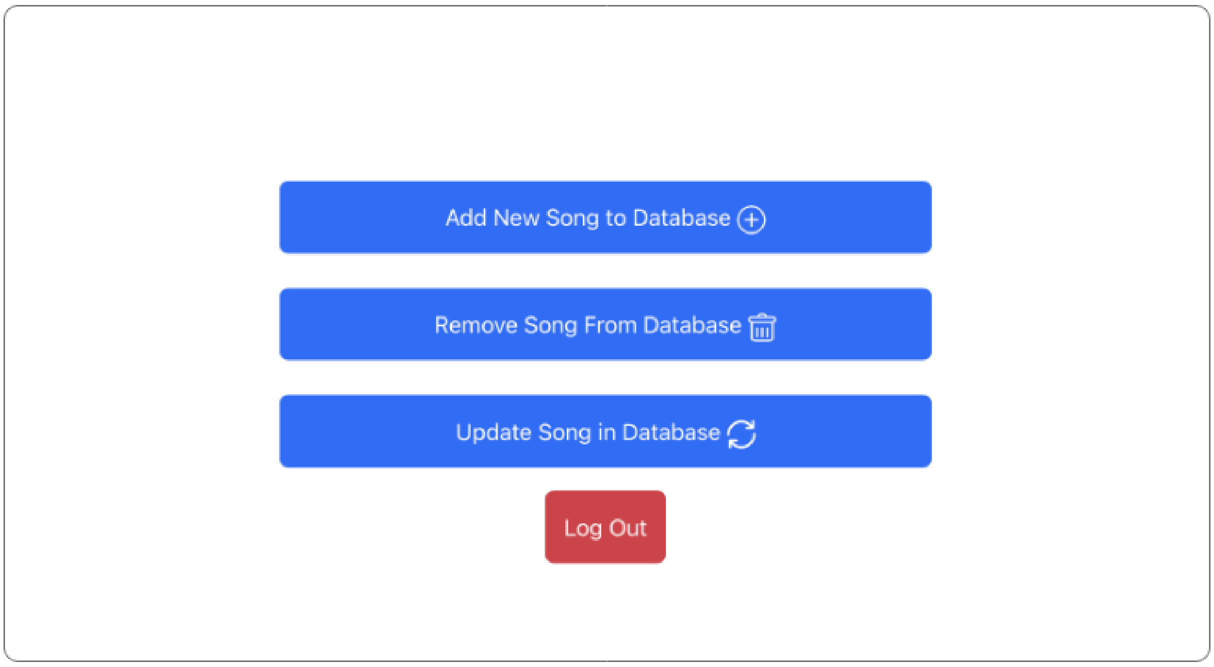


Figure 18: Admin Page (authorized)

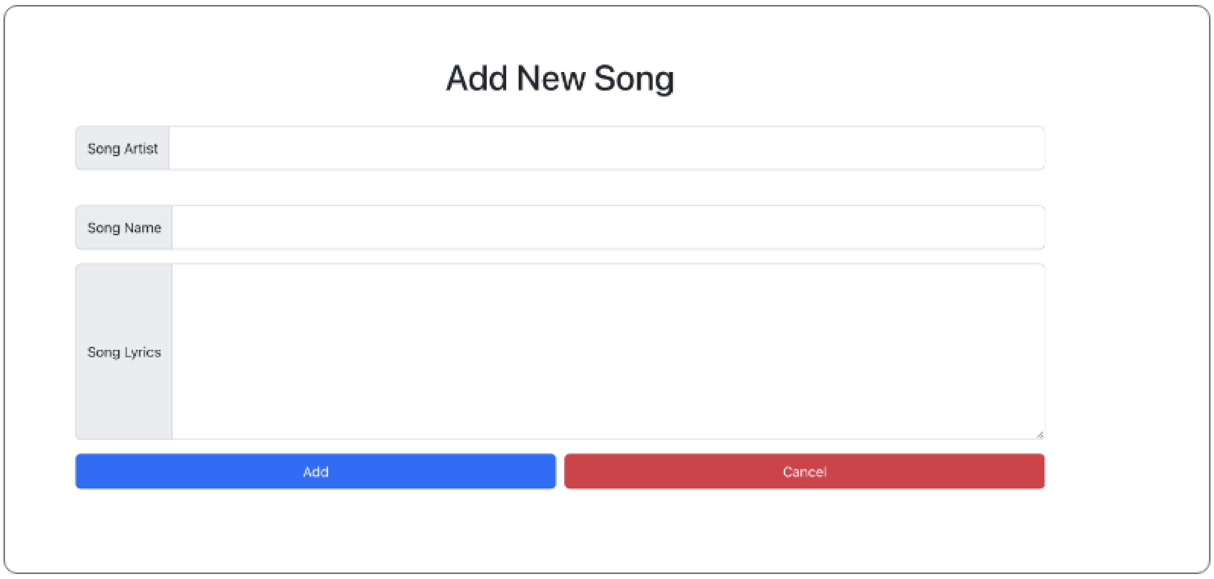
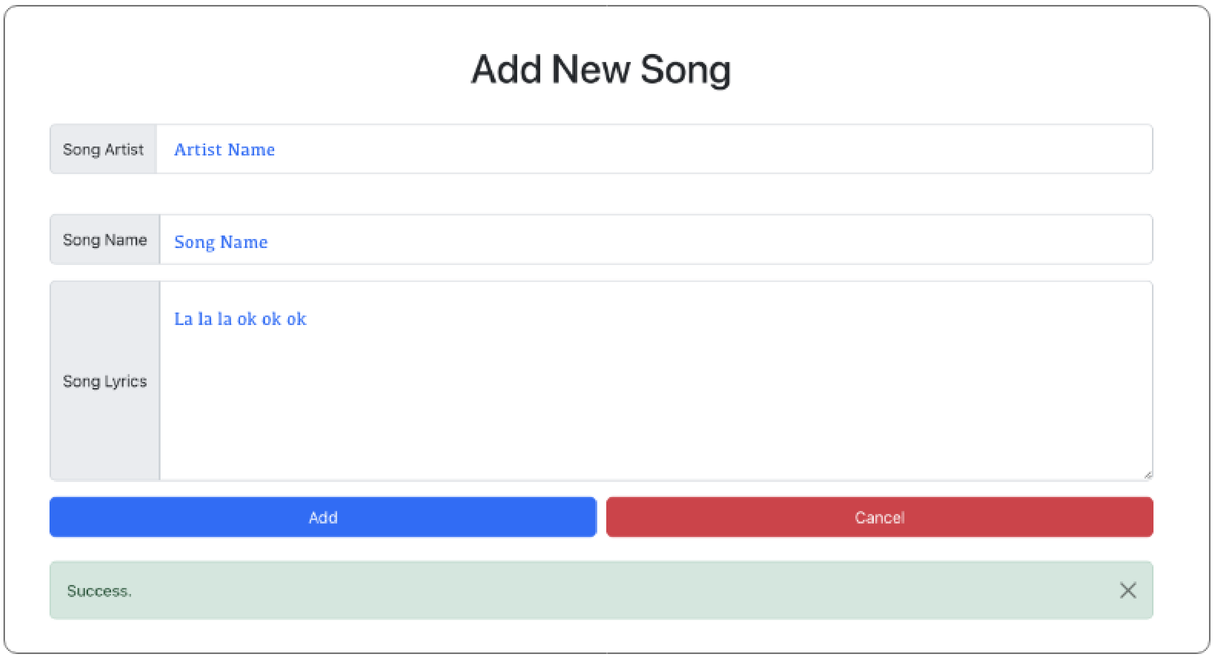


Figure 19: Admin - Add New Song to Database



Artist Name

Song Name

La la la ok ok ok

Success.

Figure 20: Admin - Add New Song to Database (success)

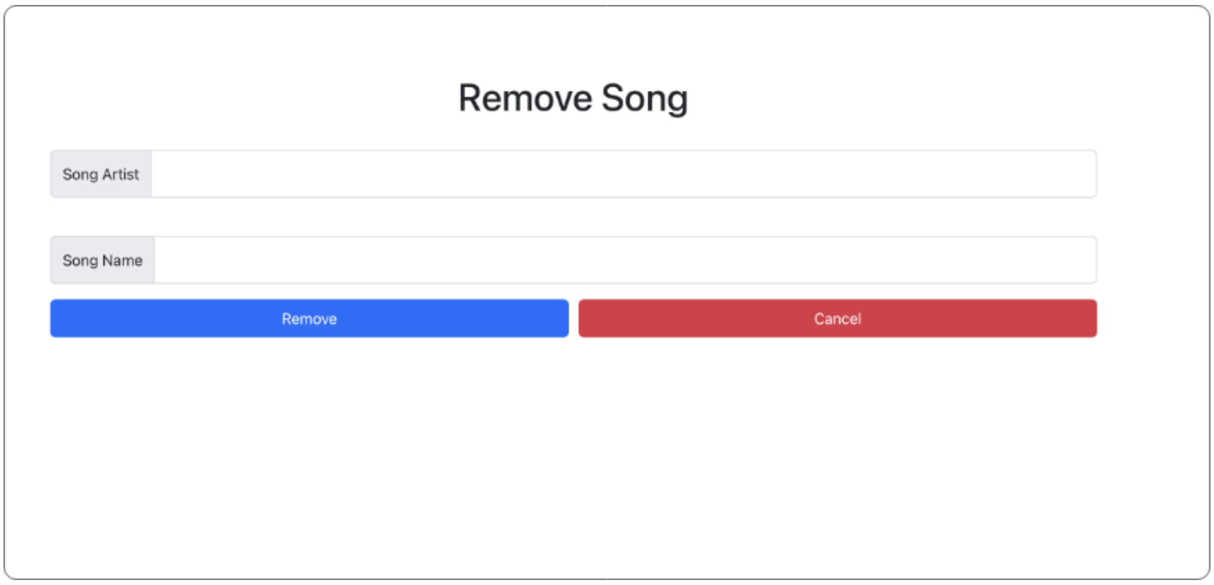
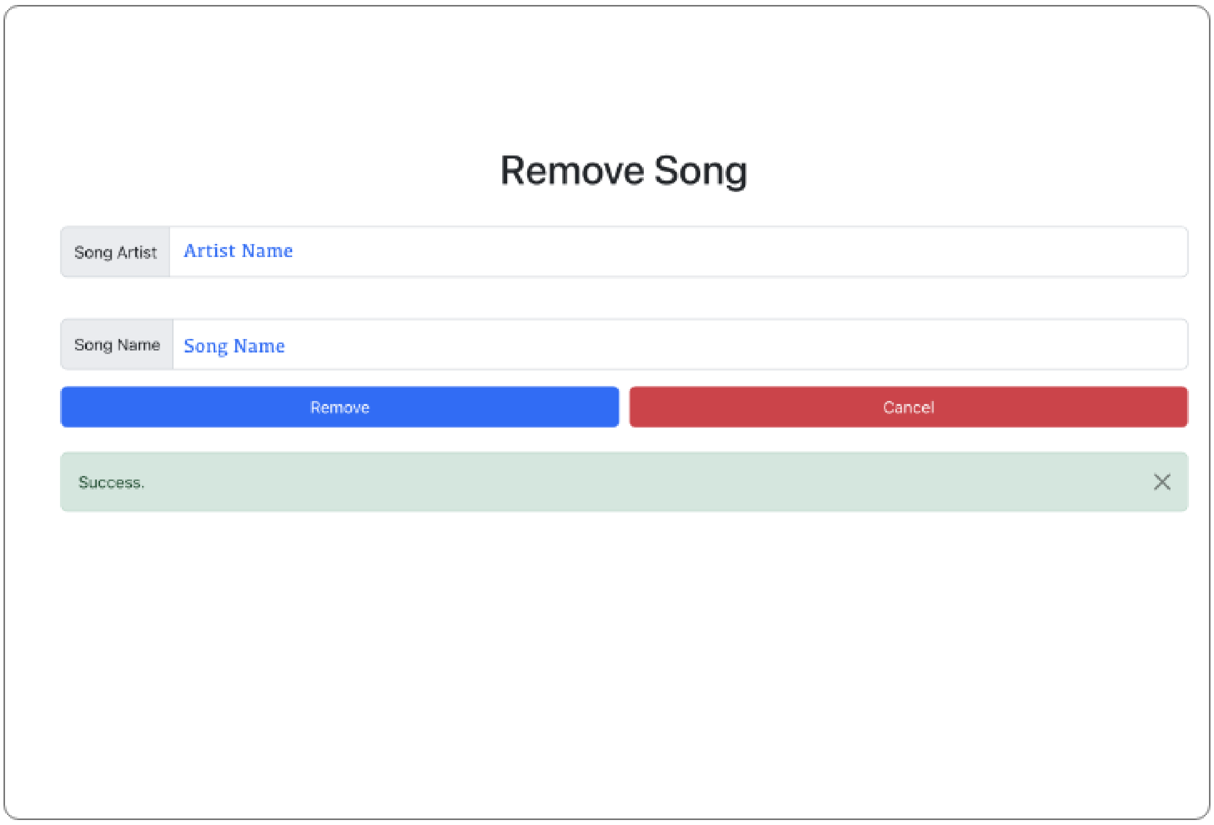


Figure 21: Admin - Remove Song from Database



Success.

Figure 22: Admin - Remove Song from Database (success)

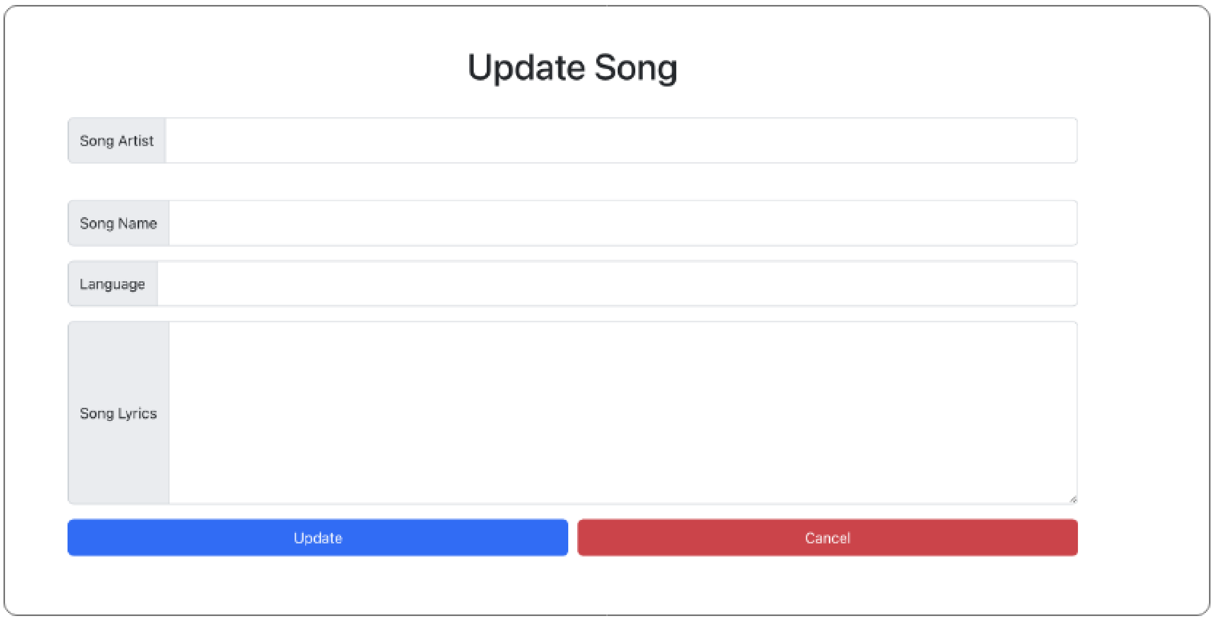
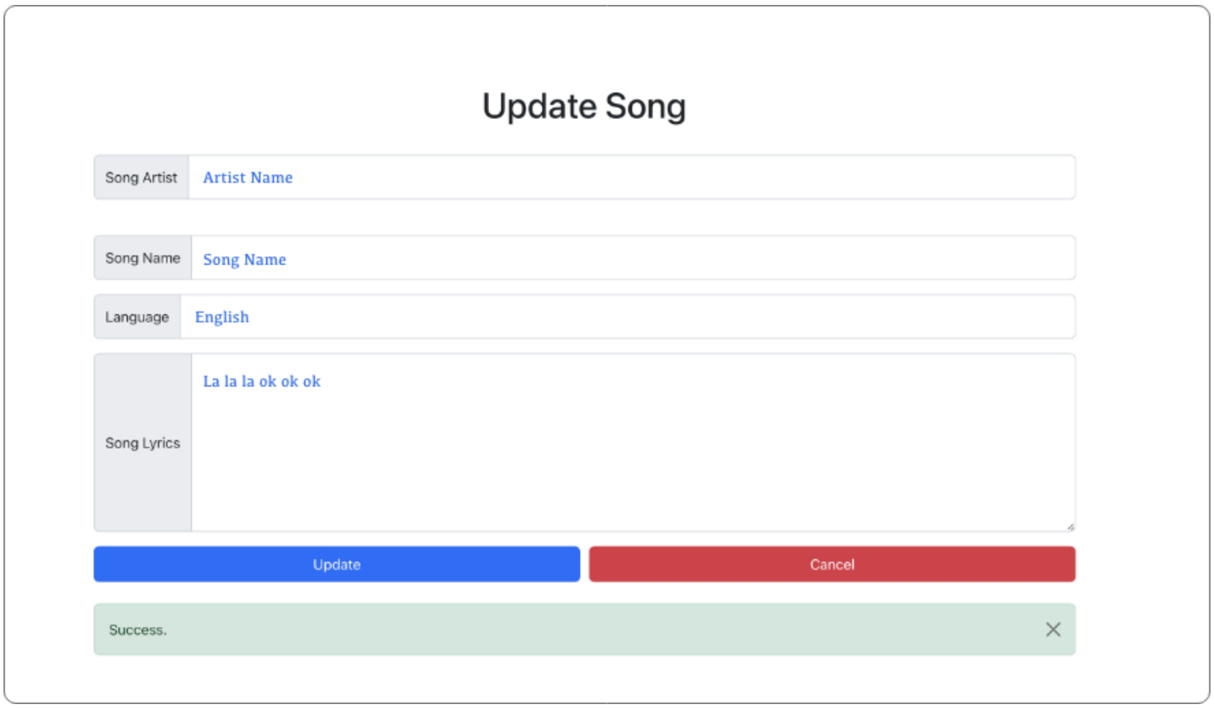


Figure 23: Admin - Update Song in Database

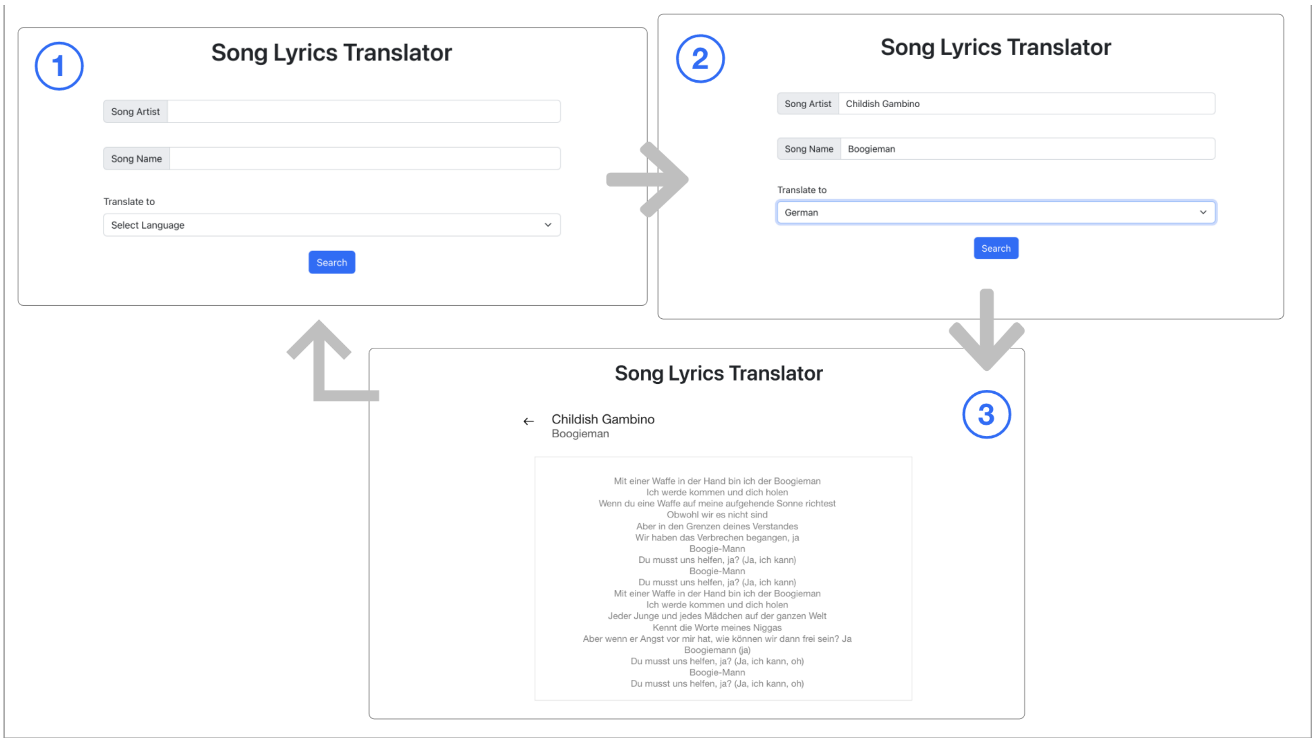


Song Artist	Artist Name
Song Name	Song Name
Language	English
Song Lyrics	La la la ok ok ok

Success.

Figure 24: Admin - Update Song in Database (success)

6.2 User Interface Navigation Flow

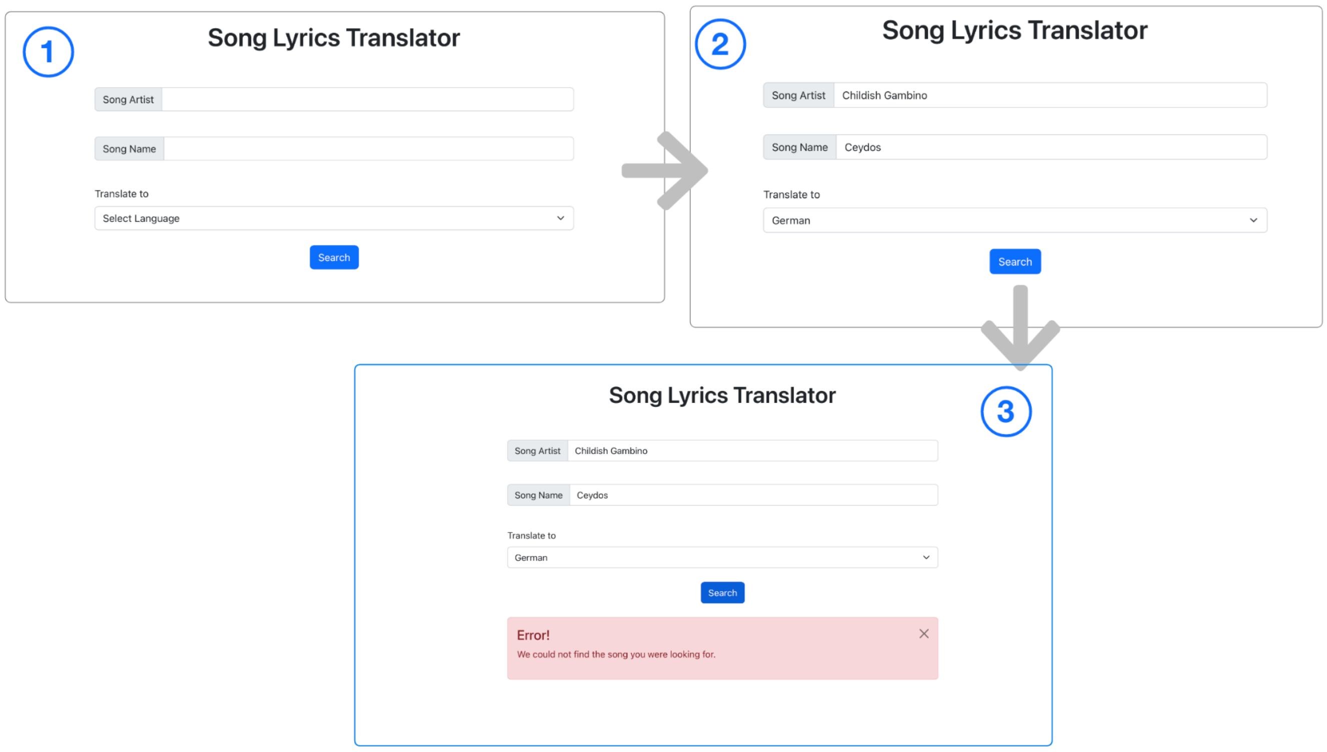


Song Artist: Childish Gambino
Song Name: Boogieman
Translate to: German

← Childish Gambino
Boogieman

Mit einer Waffe in der Hand bin ich der Boogieman
Ich werde kommen und dich holen
Wenn du eine Waffe auf meine aufgehende Sonne richtest
Obwohl wir es nicht sind
Aber in den Grenzen deines Verstandes
Wir haben das Verbrechen begangen, ja
Boogie-Mann
Du musst uns helfen, ja? (Ja, ich kann)
Boogie-Mann
Du musst uns helfen, ja? (Ja, ich kann)
Mit einer Waffe in der Hand bin ich der Boogieman
Ich werde kommen und dich holen
Jeder Junge und jedes Mädchen auf der ganzen Welt
Kennt die Worte meines Niggas
Aber wenn er Angst vor mir hat, wie können wir dann frei sein? Ja
Boogiemann (ja)
Du musst uns helfen, ja? (Ja, ich kann, oh)
Boogie-Mann
Du musst uns helfen, ja? (Ja, ich kann, oh)

Figure 25: Navigation Flow (Successful case: Song present in the database.)



Song Artist: Childish Gambino
Song Name: Ceydos
Translate to: German

Error!
We could not find the song you were looking for.

Figure 26: Navigation Flow (Failed case: Song not present in the database.)

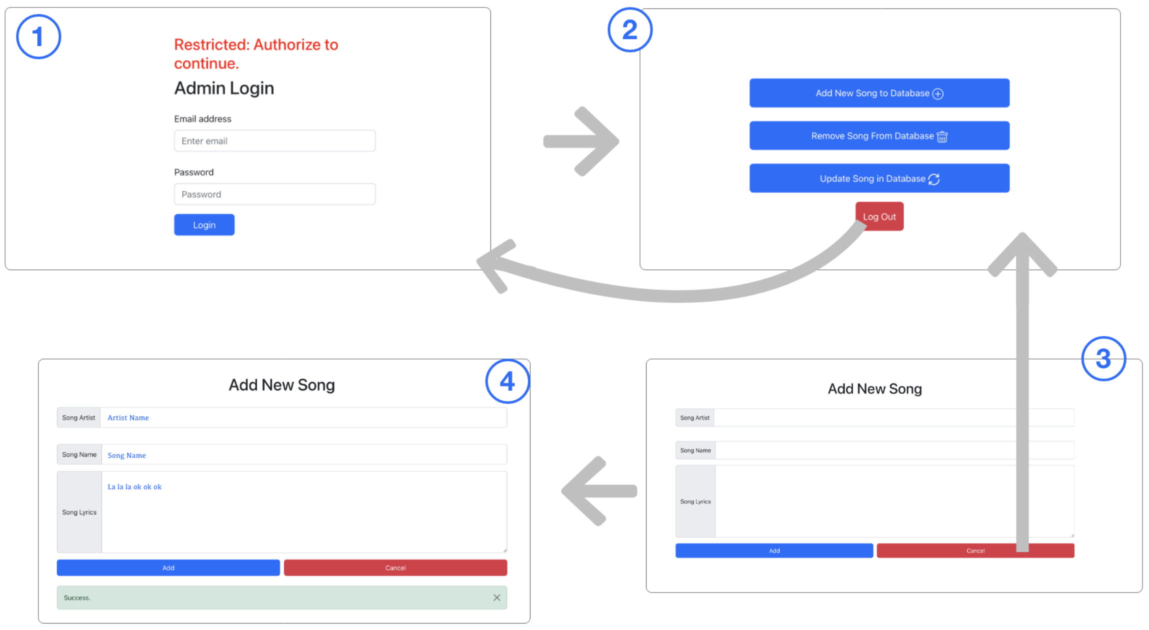


Figure 27: Navigation Flow (Admin: Add Song to Database).

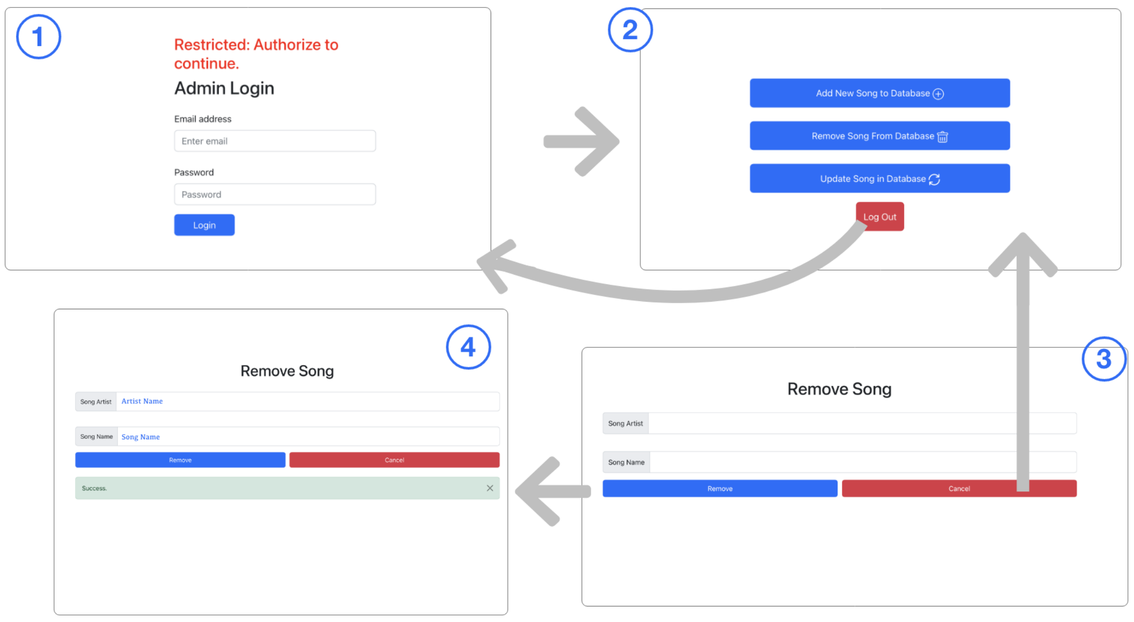
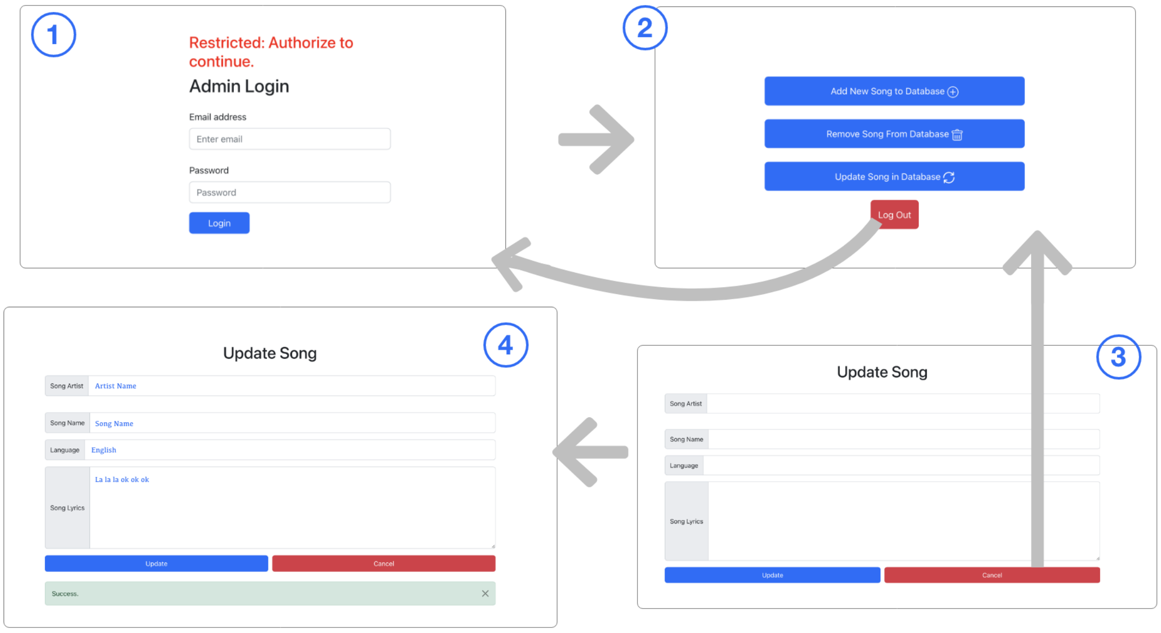


Figure 28: Navigation Flow (Admin: Remove Song from Database).



Artist Name

Song Name

English

La la la ok ok ok

Success.

Figure 29: Navigation Flow (Admin: Update Song in Database).

6.2 Use Cases / User Function Description

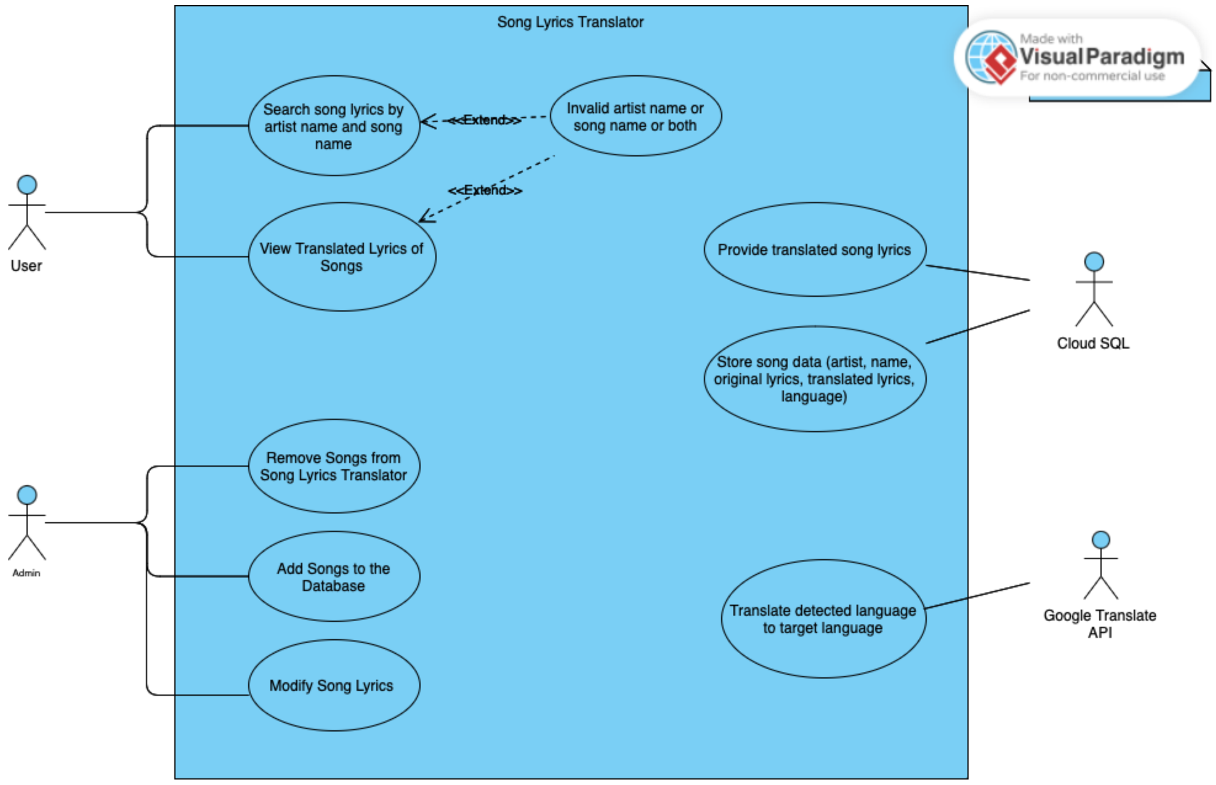

Song Lyrics Translator
Search song lyrics by artist name and song name
<<Extend>>
Invalid artist name or song name or both
<<Extend>>
View Translated Lyrics of Songs
Provide translated song lyrics
Store song data (artist, name, original lyrics, translated lyrics, language)
Remove Songs from Song Lyrics Translator
Add Songs to the Database
Modify Song Lyrics
Translate detected language to target language
User
Admin
Cloud SQL
Google Translate API
Made with Visual Paradigm For non-commercial use

Figure 30: Use Case Diagram with Internal and External Actors for Song Lyrics Translator

7. Alternatives Considered

We thought of alternatives both as a design and as features that we can add to the project:

7.1 Alternative 1 - Kubernetes

We use Google App Engine in our project. So we preferred PaaS as a cloud service type. However, we could have preferred to do this project using Google Kubernetes Engine (GKE). Thus, we could write the code in our operating system and easily transfer it to another operating system with container technology. However, Google Kubernetes Engine has some difficulties as well as such facilitating features:

- Google Kubernetes Engine requires more technical expertise to set up and manage the project.
- When compared to App Engine, GKE is more costly [2].

For these reasons, we eliminated this alternative, as we considered ourselves not very technically familiar with GKE.

7.2 Alternative 2 - Cloud Run

Similarly, we thought of using Cloud Run instead of Google App Engine, but although we did research on container technology at the beginning of the term, we did not think about this alternative too much because we did not have much knowledge in terms of applying containerization. It could have been considered at this stage of the project, but since we started to implement the project on App Engine, we decided to continue with Google App Engine.

7.3 Alternative 3 - Spring Framework

At the beginning of the project, we were thinking of writing the backend part using Spring Framework (Java). Although we are familiar with writing in Java, we had serious difficulties while connecting to the cloud. Moreover, we couldn't connect. As Node.js is easier to connect to the cloud, we gave up on the familiar framework. Therefore, we made a trade-off between familiarity with a framework and ease of connecting to the cloud.

7.4 Alternative 4 - Database

In our current model, we store both song data and translatedSong data. When a user requests translated song lyrics, it is first checked if it's present in the database (Cloud SQL). An alternative model we had to consider was obtaining translatedSong from the API every time the user requested it, without storing the translated lyrics in our database. However, charges are scaled to the number of characters provided to Cloud Translation [3], whereas charges are scaled hourly/monthly for Cloud SQL [4], and "Always Free Usage" is available for Cloud Storage [5]. Since we have to rent Cloud SQL to store and access song data nevertheless, we wanted to reduce costs by storing translated lyrics data in the cloud as well. This way, our application depends less on the availability of the Translation API.

The main concern for such an application is storing large data. Since there are more than 5M songs in our source dataset, there will be ~5M song instances with lyrics info stored, as well as more than 5M translatedSong instances (for each targetLanguage, a separate instance is stored). Considering this, we had to explore different database options that Google Cloud had to offer. We focused on relational options (Cloud SQL, Spanner, BigQuery, etc.) rather than non-relational options such as Firestore and Bigtable [6]. We eliminated Cloud Spanner since it is one of the expensive options, and it is recommended for global-scale applications, which is overkill for our application. BigQuery, on the other hand, is more suited for analytics. We decided that Cloud SQL would be more than sufficient for our simple data design (see Tables 1 and 2).

7.5 Feature Alternative - Login

In an app like Song Lyrics Translator, we considered an alternative where we had a "user login" function where we stored user data along with static song data. However, for better user experience (easier to access the main feature without logging in) and the simplicity of our application, our decision was to omit that feature. Thus, a simple data model came out for this project with only 3 types of tables: song, translatedSong, and users.

8. Testing Plan

We performed unit tests on each module and function. By doing that we ensured that every section of the code is working as it should be. After performing unit tests, we combined everything together. Additionally, we evaluated the system's performance under various situations, including edge cases and boundary conditions. In some parts, we used Postman to test if our backend is working properly. After we ensured that our backend works properly, we performed some test cases. The test cases we have carried out for the project are as follows (including Postman tests):

Test No: 1 - Postman	**Test Result**: Successful
Test Title: Testing whether the backend is working or not	
Test Steps: ● Open Postman ● Set method to "GET" ● Enter the following link: https://server-dot-hazel-env-380416.lm.r.appspot.com/ ● Click the "Send" Button	
Expected Result: It should print "Hello Backend"	

Test No: 2 - Postman	**Test Result**: Successful
Test Title: Testing Backend - Database connection	
Test Steps: ● Open Postman ● Set method to "GET" ● Enter the following link: https://server-dot-hazel-env-380416.lm.r.appspot.com/songs ● Click the "Send" Button	
Expected Result: It should print the top two songs in the "song" table.	

Test No: 3 - Postman **Test Result**: Successful

Test Title: Testing backend song translation

Test Steps:
- Open Postman
- Set method to “GET”
- Enter the following link: https://server-dot-hazel-env-380416.lm.r.appspot.com/get-song-info?artistName=Celine+Dion&songName=My+Heart+Will+Go+On&desiredLanguage=tr
- Click the “Send” Button

Expected Result: It should return the original and translated lyrics of Celine Dion's song.

Test No: 4 **Test Result**: Successful

Test Title: The user wants to translate a song that is in the database

Test Steps:
- Go to the main page.
- Enter input with a compatible song and artist name.
  (For example, Song Artist: Celine Dion
  Song Name: My Heart Will Go On)
- Choose the language we want to translate. (For example, Korean)
- Click the “Search” button.

Expected Result: It should redirect us to the page where the original and translated versions of the lyrics are available.

Test No: 5 **Test Result**: Successful

Test Title: The user wants to translate a song that **is not** in the database

Test Steps:
- Go to the main page.
- Enter input with incompatible song and artist name.
  (For example, Song Artist: Cloud
  Song Name: Cloud)
- Choose the language we want to translate. (For example, Chinese)
- Click the “Search” button.

Expected Result: It should give an error message that the song is not found in the database.

Test No: 6	**Test Result**: Successful
Test Title: Accessing the /home extension directly with no credentials	
Test Steps: ● Go to the main page. ● Write /home at the end of the website address	
Expected Result: It should redirect to /admin extension because we don’t have the right credentials.	

Test No: 7	**Test Result**: Successful
Test Title: Trying to login to the admin page with the **wrong** credentials	
Test Steps: ● Go to the main page. ● Write /admin at the end of the website address. ● Enter input with a false username and password. (For example, Username: admin Password: admin) ● Click the “Login” button	
Expected Result: It should give an error message saying that we are trying to login with incorrect credentials.	

Test No: 8	**Test Result**: Successful
Test Title: Trying to login to the admin page with **valid** credentials	
Test Steps: ● Go to the main page. ● Write /admin at the end of the website address. ● Enter input with true username and password. (For example, Username: testuser Password: testpassword) ● Click the “Login” button	
Expected Result: It should generate a new token and redirect us to the admin control page.	

Test No: 9	**Test Result**: Successful
Test Title: Trying to access the /home extension while the token exists and then logging out.	
Test Steps: ● Go to the main page. ● Write /home at the end of the website address. ● Click the “Logout” button	
Expected Result: We should be able to access the /home extension while the token exists, and our token should be lost when logging out.	

Test No: 10	**Test Result**: Successful
Test Title: Trying to add a new song	
Test Steps: ● Go to the main page. ● Write /admin at the end of the website address. ● Enter input with true username and password. (For example, Username: testuser Password: testpassword) ● Click the “Login” button ● Click the “Add New Song to Database” ● Enter input with some song artist, name, and lyrics. ○ (For example, Song Artist: Rihanna Song Name: Diamonds Song Lyrics: Hello World) ● Click the “Add” button.	
Expected Result: It should give an alert on the screen that adding the song to the database was successful.	

Test No: 11	**Test Result**: Successful
Test Title: Trying to remove a song	
Test Steps: ● Go to the main page. ● Write /admin at the end of the website address. ● Enter input with true username and password. (For example, Username: testuser Password: testpassword) ● Click the “Login” button ● Click the “Remove Song From Database” ● Enter input with some song artist and name. ○ (For example, Song Artist: Rihanna Song Name: Diamonds) ● Click the “Remove” button.	

Expected Result: It should give an alert on the screen that it has removed the song from the database.

Test No: 12	**Test Result**: Successful
Test Title: Trying to update a song lyrics	
Test Steps: ● Go to the main page. ● Enter input with a compatible song and artist name. (For example, Song Artist: Celine Dion Song Name: My Heart Will Go On) ● Choose the language we want to translate. (For example, Turkish) ● Copy the Turkish-translated version of the lyrics. ● Write /admin at the end of the website address. ● Enter input with true username and password. (For example, Username: testuser Password: testpassword) ● Click the "Login" button. ● Click the "Update Song in Database". ● Enter input with some song artist, name, language, and lyrics. For the Celine Dion example, use: Song Artist: Celine Dion Song Name: My Heart Will Go On Language: tr Lyrics: (Paste copied lyrics and change some words) ● Click the "Update" button.	
Expected Result: It should show an alert that the lyrics have been successfully modified in the desired language.	

9. Demo Plan

The use case we are planning to demonstrate in our final demo is for a user who wants to translate song lyrics from one language to another. The user will enter the target language, artist, and song title into the web application's front-end server. The back-end server will process the user's input and use Cloud Storage to get the original lyrics for the requested song. The original lyrics will be translated using the Translation API and placed in Cloud SQL if the translated lyrics are not already there in the database. The user will then receive the translated lyrics in the chosen language.

We will demonstrate how the app can process user input, use Cloud Storage and Cloud SQL, and connect to the Translation API to produce precise and meaningful translations. We will use sample songs and languages to illustrate the translation process, and we will check the returned results. We'll also demonstrate the application's user interface and how it enables users to input data and get the translated lyrics.

We will also show the admin who wants to add/remove/modify songs.

The steps we implemented in the demo are as follows:

1- The user clicks on song search without entering any input. The system will ask you to fill in the blank fields in the input section.

2- After that the user tries to search for a song that is not presented in the database. The system will give an alert that states the requested song is not found in the database.

3- Then we showed our “song” table and got a song from there. We translated this song’s lyrics (Celine Dion - My Heart Will Go On) into Turkish. After that, we showed the "translatedSongs" table to show that the Turkish lyrics of the song are kept in the database.

4- We moved the admin scenario and tried to enter the admin page with the wrong credential in the admin login page. It will give an error because of the wrong credentials.

5- After that, we logged in with the correct credentials, showed the existence of the token, and explained that the page cannot be accessed without the token.

6- We tried adding a song (Rihanna - Diamonds) as an admin. After adding, we showed the song was added successfully to the database.

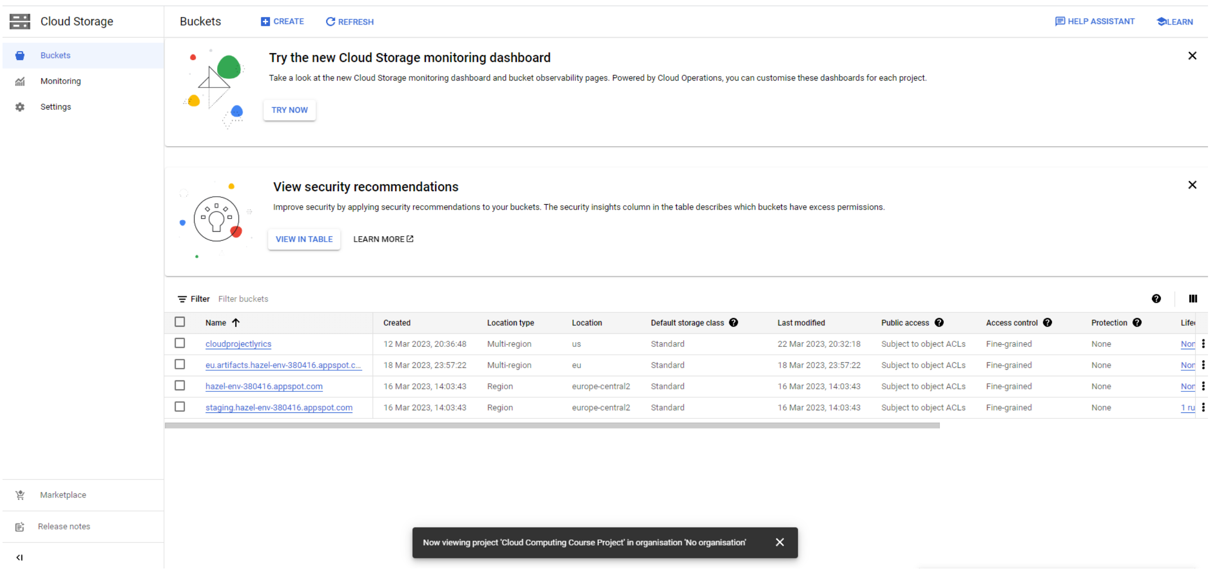
7- We switched to the user side and translated the recently added song into Turkish. We showed that the Turkish version of this song is also in "translatedSongs".

8- We went to the admin side and changed the lyrics of the Turkish version of this song. We went to the User section and checked the lyrics again and showed that we had successfully modified the lyrics. Later, we showed that it has also changed in the "translatedSongs" table.

9- We deleted the last song we added (Rihanna - Diamonds) and showed that it was also deleted on the necessary tables.

Demo Link:
https://drive.google.com/file/d/1HiqrEYiezz1fDaks9vWC8qFFtHP1U_SB/view?usp=sharing

10. Cloud Setup



Try the new Cloud Storage monitoring dashboard

Take a look at the new Cloud Storage monitoring dashboard and bucket observability pages. Powered by Cloud Operations, you can customise these dashboards for each project.

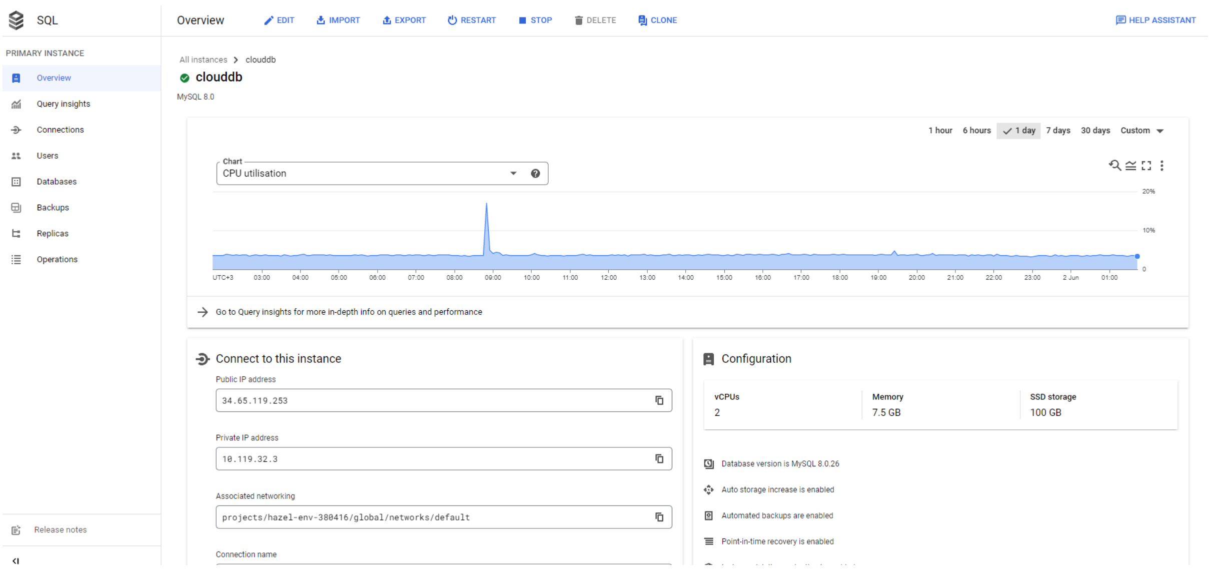
View security recommendations

Improve security by applying security recommendations to your buckets. The security insights column in the table describes which buckets have excess permissions.

Name	Created	Location type	Location	Default storage class	Last modified	Public access	Access control	Protection	Life
cloudprojectlyrics	12 Mar 2023, 20:36:48	Multi-region	us	Standard	22 Mar 2023, 20:32:18	Subject to object ACLs	Fine-grained	None	Non
eu.artifacts.hazel-env-380416.appspot.c...	18 Mar 2023, 23:57:22	Multi-region	eu	Standard	18 Mar 2023, 23:57:22	Subject to object ACLs	Fine-grained	None	Non
hazel-env-380416.appspot.com	16 Mar 2023, 14:03:43	Region	europe-central2	Standard	16 Mar 2023, 14:03:43	Subject to object ACLs	Fine-grained	None	Non
staging.hazel-env-380416.appspot.com	16 Mar 2023, 14:03:43	Region	europe-central2	Standard	16 Mar 2023, 14:03:43	Subject to object ACLs	Fine-grained	None	1 ru

Now viewing project 'Cloud Computing Course Project' in organisation 'No organisation'

Figure 31: Downloaded dataset in Cloud Storage



All instances > clouddb

clouddb

MySQL 8.0

Chart: CPU utilisation

Go to Query insights for more in-depth info on queries and performance

Connect to this instance

Public IP address: `34.65.119.253`

Private IP address: `10.119.32.3`

Associated networking: `projects/hazel-env-380416/global/networks/default`

Connection name

Configuration

vCPUs	Memory	SSD storage
2	7.5 GB	100 GB

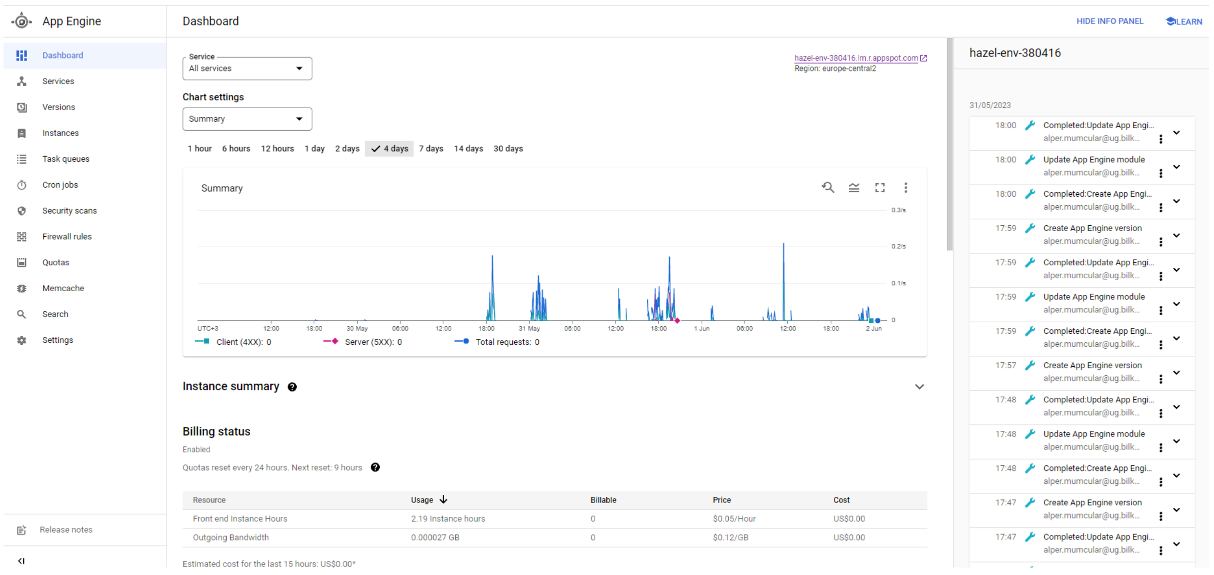
Database version is MySQL 8.0.26

Auto storage increase is enabled

Automated backups are enabled

Point-in-time recovery is enabled

Figure 32: Cloud SQL - MySQL instance



Dashboard

Service: All services

Chart settings: Summary

hazel-env-380416.lm.r.appspot.com
Region: europe-central2

🟦 Client (4XX): 0 — ◆ Server (5XX): 0 — ● Total requests: 0

Instance summary

Billing status

Enabled

Quotas reset every 24 hours. Next reset: 9 hours

Resource	Usage	Billable	Price	Cost
Front end Instance Hours	2.19 Instance hours	0	$0.05/Hour	US$0.00
Outgoing Bandwidth	0.000027 GB	0	$0.12/GB	US$0.00

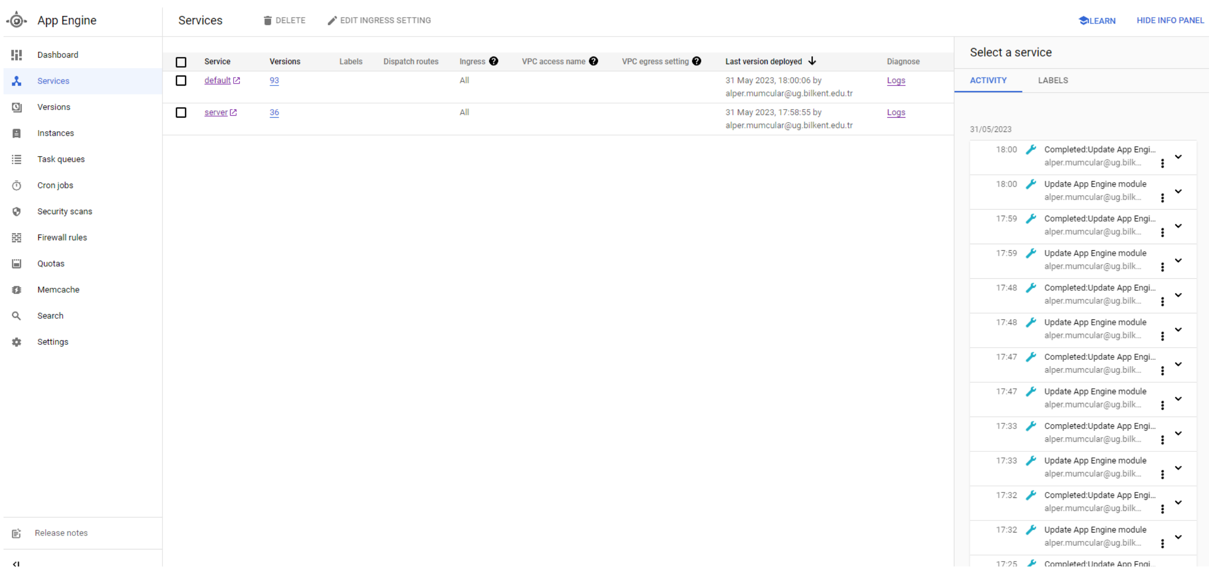
Estimated cost for the last 15 hours: US$0.00*

hazel-env-380416

31/05/2023

```
18:00  Completed:Update App Engi...   alper.mumcular@ug.bilk...
18:00  Update App Engine module       alper.mumcular@ug.bilk...
18:00  Completed:Create App Engi...   alper.mumcular@ug.bilk...
17:59  Create App Engine version      alper.mumcular@ug.bilk...
17:59  Completed:Update App Engi...   alper.mumcular@ug.bilk...
17:59  Update App Engine module       alper.mumcular@ug.bilk...
17:59  Completed:Create App Engi...   alper.mumcular@ug.bilk...
17:57  Create App Engine version      alper.mumcular@ug.bilk...
17:48  Completed:Update App Engi...   alper.mumcular@ug.bilk...
17:48  Update App Engine module       alper.mumcular@ug.bilk...
17:48  Completed:Create App Engi...   alper.mumcular@ug.bilk...
17:47  Create App Engine version      alper.mumcular@ug.bilk...
17:47  Completed:Update App Engi...   alper.mumcular@ug.bilk...
```

Figure 33: App Engine - Dashboard



Services

Service	Versions	Labels	Dispatch routes	Ingress	VPC access name	VPC egress setting	Last version deployed	Diagnose
default	93			All			31 May 2023, 18:00:06 by alper.mumcular@ug.bilkent.edu.tr	Logs
server	36			All			31 May 2023, 17:58:55 by alper.mumcular@ug.bilkent.edu.tr	Logs

Select a service

31/05/2023

```
18:00  Completed:Update App Engi...   alper.mumcular@ug.bilk...
18:00  Update App Engine module       alper.mumcular@ug.bilk...
17:59  Completed:Update App Engi...   alper.mumcular@ug.bilk...
17:59  Update App Engine module       alper.mumcular@ug.bilk...
17:48  Completed:Update App Engi...   alper.mumcular@ug.bilk...
17:48  Update App Engine module       alper.mumcular@ug.bilk...
17:47  Completed:Update App Engi...   alper.mumcular@ug.bilk...
17:47  Update App Engine module       alper.mumcular@ug.bilk...
17:33  Completed:Update App Engi...   alper.mumcular@ug.bilk...
17:33  Update App Engine module       alper.mumcular@ug.bilk...
17:32  Completed:Update App Engi...   alper.mumcular@ug.bilk...
17:32  Update App Engine module       alper.mumcular@ug.bilk...
17:25  Completed:Update App Engi...
```

Figure 34: App Engine - Services

11. References

[1] “Genius Song Lyrics | Kaggle.” https://www.kaggle.com/datasets/carlosgdcj/genius-song-lyrics-with-language-information (accessed Apr. 08, 2023).

[2] “Google Kubernetes Engine (GKE),” *Google Cloud*. https://cloud.google.com/kubernetes-engine (accessed Apr. 09, 2023).

[3] “Pricing | Cloud Translation | Google Cloud.” https://cloud.google.com/translate/pricing (accessed Mar. 30, 2023).

[4] “Pricing | Cloud SQL: Relational Database Service,” *Google Cloud*. https://cloud.google.com/sql/pricing (accessed Mar. 30, 2023).

[5] “Pricing | Cloud Storage | Google Cloud.” https://cloud.google.com/storage/pricing (accessed Mar. 30, 2023).

[6] “Google Cloud Databases,” *Google Cloud*. https://cloud.google.com/products/databases (accessed Apr. 09, 2023).