Bilkent University

Department of Computer Science

CS353 DATABASE SYSTEMS

Sudo: Music listening and sharing platform

Final Report

Doruk Çakmakçı
Hakan Sarp Aydemir
Mehmet Oğuz Göçmen
Umut Berk Bilgiç

Course Instructor: Abdullah Ercüment Çiçek

14.5.2018

Table of Contents

1. Project Description

Sudo is a web-based application for publishing and reaching to people with music as well as listening to music and commenting about it, similar to “Spotify”. This application is in a sense a social platform which is built with the purpose of sharing music with people all around the world. To achieve this purpose, one needs a place that is enough to store all the information of existing accounts, keep track of accounts that is registered to system and all their public and private information such as their biographies or passwords etc. Moreover, since users will be able to comment on many tracks and share them with their friends, existing data must be clear enough to be manipulated. To control all these vast information, the need for a database becomes more necessary. To achieve all the desired needs, the database should be connected enough to provide these services to the users of the system.

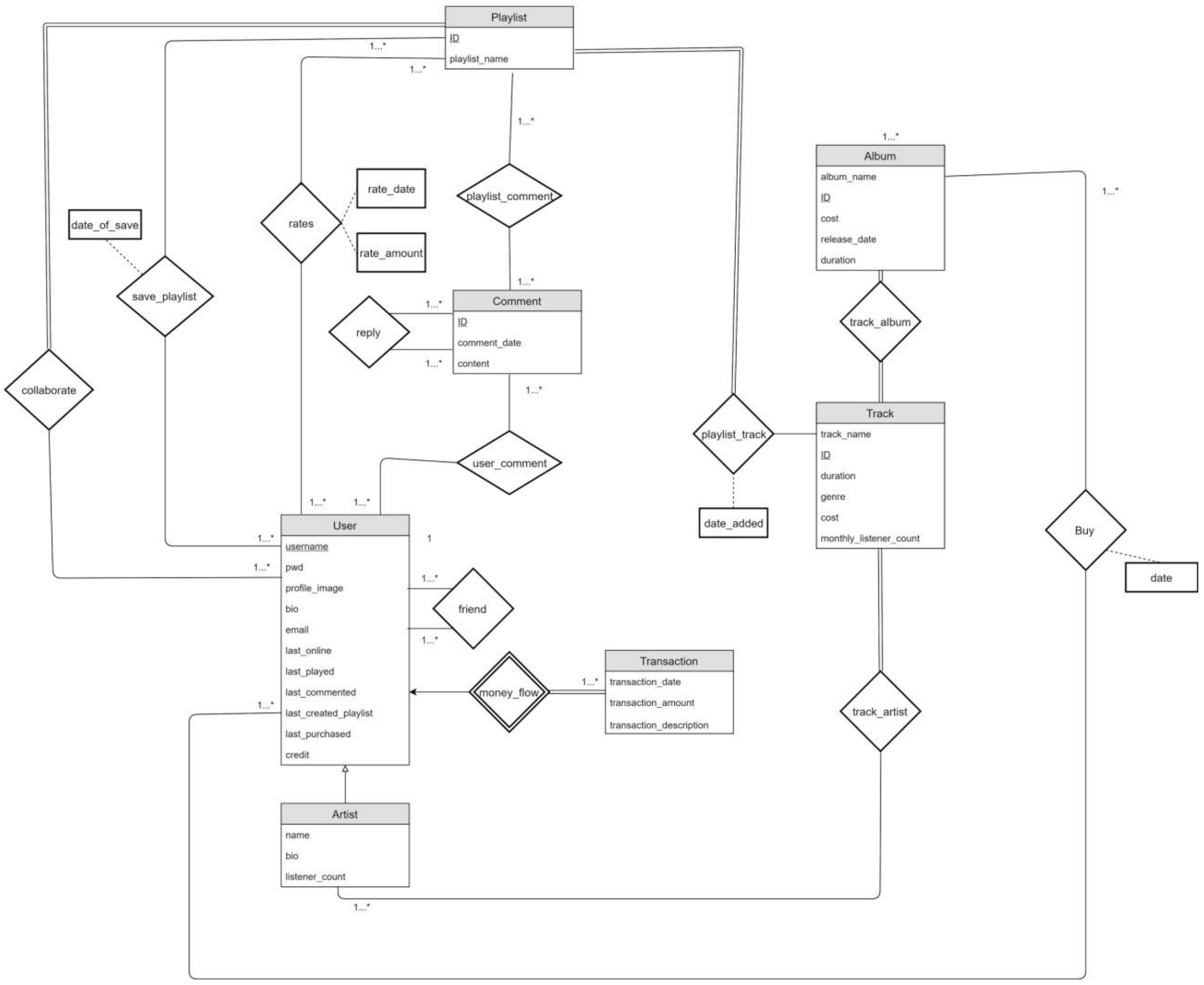
The system contains several entities such as artists, tracks, albums, playlists and users who are able to create their own playlists, buy songs, comment and share the tracks they listen to. Albums are either consisted of several tracks or just a single track. Playlists contain one’s favorite tracks that he/she added into.

In our application there are two main type of users; User and Artist. Users are able to search tracks, albums and artists and find the ones that are best suitable to their music taste. They are able to add friends, comment and share their playlists, also they can create playlists collaboratively.

Artists are able to publish their brand-new tracks and share information about these tracks as well as their own biography. This published product can be either a single or an album. Their tracks can be bought by users; so buy count of their tracks will also be displayed in their profile. Artist can also do everything a normal user can do.

Doruk worked completely on back-end and arranged connections between database and website using MySQL & PHP. Umut worked mostly on front-end development using BootStrap, HTML & Javascript; also helped with some PHP scripts . Sarp worked on both front-end and back-end development using HTML/CSS/BootStrap on front-end, PHP on back-end. Oğuz also worked on both front-end and back-end development using HTML/CSS/BootStrap on front-end, PHP on back-end. All members worked on MySQL queries that we use in database.

2. Final E/R Diagram



Final E/R Diagram

3. Relation Schemas

3.1. User

User(username, pwd, profile_image, bio, email, last_online, last_played, last_comment, last_created_playlist, last_purchased, credit)

FOREIGN KEY: last_played **REFERENCES** Track(ID)

FOREIGN KEY: last_comment **REFERENCES** Comment(ID)

FOREIGN KEY: last_created_playlist **REFERENCES** Playlist(ID)

FOREIGN KEY: last_purchased **REFERENCES** Album(ID)

3.2. Artist

Artist(username, artist_name, bio, listener_count)

FOREIGN KEY: username **REFERENCES** User(username)

3.3. Friend

Friend(follower_ID, following_ID)

FOREIGN KEY: follower_ID **REFERENCES** User(username)

FOREIGN KEY: following_ID **REFERENCES** User(username)

3.4. Track

Track(ID, track_name, track_duration, genre, track_cost, monthly_listener_count)

3.5. Album

Album(ID, album_name, album_cost, album_cover, album_duration,artist_username)

FOREIGN KEY: artist_username **REFERENCES** Artist(username)

3.6. Track_Album

Track_Album(album_ID, track_ID)

FOREIGN KEY: album_ID **REFERENCES** Album(ID)

FOREIGN KEY: track_ID **REFERENCES** Track(ID)

3.7. Track_Artist

Track_Artist(track_ID, artist_username)

FOREIGN KEY: track_ID **REFERENCES** Track(ID)

FOREIGN KEY: artist_username **REFERENCES** Artist(username)

3.8. Playlist

Playlist(playlist_ID, playlist_name, creator_username, avg_rating, creation_date)

FOREIGN KEY: creator_username **REFERENCES** User(username)

3.9. Playlist_Track

Playlist_Track(playlist_ID, track_ID, date_added)

FOREIGN KEY: playlist_ID **REFERENCES** Playlist(ID)

FOREIGN KEY: track_ID **REFERENCES** Track(ID)

3.10. Save_Playlist

Save_Playlist(playlist_ID, username, date_of_save)

FOREIGN KEY: playlist_ID **REFERENCES** Playlist(ID)

FOREIGN KEY: username **REFERENCES** User(username)

3.11. Collaborate

Collaborate(username, playlist_ID)

FOREIGN KEY: username **REFERENCES** User(username)

FOREIGN KEY: playlist_ID **REFERENCES** Playlist(ID)

3.12. Rates

rates(playlist_ID, username, rate_date, rate_amount)

FOREIGN KEY: playlist_ID **REFERENCES** to playlist(ID)

FOREIGN KEY: username **REFERENCES** User(username)

3.13. Comment

Comment(comment_ID, comment_date, comment_content)

3.14. Reply

Reply(parent_ID, child_ID)

FOREIGN KEY: parent_ID **REFERENCES** Comment(ID)

FOREIGN KEY: child_ID **REFERENCES** Comment(ID)

3.15. Playlist_Comment

playlist_comment(playlist_ID, comment_ID)

FOREIGN KEY: playlist_ID **REFERENCES** Playlist (ID)

FOREIGN KEY: comment_ID **REFERENCES** Comment (ID)

3.16. User_Comment

user_comment(comment_ID, username)

FOREIGN KEY: comment_ID **REFERENCES** Comment (ID)

FOREIGN KEY: username **REFERENCES** User (username)

3.17. Transaction

Transaction(username, transaction_ID, transaction_date, transaction_amount, transaction_description)

FOREIGN KEY: username **REFERENCES** User (username)

3.18. Buy

Buy(album_ID, username, purchase_date)

FOREIGN KEY: username **REFERENCES** User (username)

FOREIGN KEY: album_ID **REFERENCES** Album (ID)

4. Implementation Details

Our database management system consists of two main parts; database and website. For database implementation, we used MySQL. We first written all the queries by ourselves and tested in MySQL Workbench for their validations. Then we have imported all SQL queries into PHPMyAdmin as a single script file and manipulated further on from PHPMyAdmin.

For user interface and application functionalities, we first wrote an HTML template manually and then used a tool called Pingendo to move around and edit the contents of Bootstrap 4.0 elements. Additional PHP code was used within the HTML code to modify the webpage contents according to the session variables. Then we edited the HTML, CSS and JavaScript to further satisfy our needs for the website.

For the backend, we used PHP to handle system operations. Most of the PHP files are embedded inside HTML files since we needed to manipulate certain HTML tags. At first, we had written a "config.php" file which initializes server to connect and database that will be used; then, we opened a connection to that database and held "$con" variable to access. Moreover, we mostly used "session_start", "session_end" in each PHP file to open session and executed queries through "$con" variable that we held previously. At last, we used "echo" statements to provide new elements to HTML portion whose content come from the results of executed queries.

During implementation stage, we faced several issues about back-end development. First we have encountered "Header is already sent" error. This issue was solved by moving all the scripts that contain header information before any output is made. Then, we had an issue about the types of primary key's of playlist, track and album tables of our database, and foreign key constraints. To change the type of the primary key, the corresponding foreign key constraints were dropped. Then after type change, foreign key constraints were reintroduced in an updated manner. Also, as another problem, we needed to use $_GET superglobal in order to transfer data from one page to another page. Using $_GET caused the bootstrap to be invalid for the target page because the relative path to the folder in which we held images. To solve, we used ../../ instead of ../.

5. Advanced Database Components

5.1. Views

5.1.1. Past Transactions View

```
CREATE VIEW past_transactions (amount , description , transaction_date) AS
    SELECT T.transaction_amount, T.transaction_description, T.transaction_date
    FROM Transaction T NATURAL JOIN User S
    WHERE T.username = @username
```

5.1.2. Search View

```
CREATE VIEW search_page (artist_name , artist_bio , artist_picture , related_playlists , related_albums ,
related_tracks) AS
    SELECT A.artist_name, A.bio, U.profile_image, P.playlist_name, A.album_name,
           T.track_name
    FROM Artist A JOIN
        Album Al ON A.username = Al.artist_username
            JOIN
        User U ON A.username = U.username
            JOIN
        Playlist P ON P.creator_username = A.username
            JOIN
        track_album TA ON TA.album_ID = Al.ID
            JOIN
        Track T ON T.ID = TA.track_ID
    WHERE A.username = @username
```

5.1.3 My Playlists View

```
CREATE VIEW my_playlists (playlist_name , date_added , no_of_saves) AS
    SELECT P.playlist_name, P.creation_date, transaction_date,
        (SELECT COUNT(*)
         FROM save_playlist SP JOIN Playlist P2 ON SP.playlist_ID = P2.playlist_ID
         WHERE U.username = @username)
    FROM User U JOIN Playlist P ON U.username = P.creator_username
    WHERE U.username = @username
```

5.1.4. Friend View

```
CREATE VIEW `friends` (username, bio, last_online, last_played, last_comment,last_created_playlist,
last_purchased) AS
        SELECT username, bio, last_online, last_played, last_comment, last_created_playlist,
        last_purchased
        FROM `user
```

5.2. Triggers

5.2.1. Album_BEFORE_INSERT

```
CREATE TRIGGER `Album_BEFORE_INSERT` BEFORE INSERT ON `album`
 FOR EACH ROW BEGIN
        IF(new.cost < 0) THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'invalid album cost';
    END IF;
END
```

5.2.2. Playlist_AFTER_UPDATE

```
CREATE TRIGGER `Playlist_AFTER_UPDATE` AFTER UPDATE ON `playlist`
 FOR EACH ROW BEGIN
        IF(new.avg_rating > 10 OR new.avg_rating < 0) THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'invalid avg rating';
    END IF;
END
```

5.2.3.Rates_BEFORE_INSERT

```
CREATE TRIGGER `Rates_BEFORE_INSERT` BEFORE INSERT ON `rates`
 FOR EACH ROW BEGIN
        IF ( (rate_amount  < 0) OR (rate_amount > 10)) THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'invalid rate amount';
    END IF;
END
```

5.2.4. Track_BEFORE_UPDATE

```
CREATE TRIGGER `Track_BEFORE_UPDATE` BEFORE UPDATE ON `track`
 FOR EACH ROW BEGIN
        IF(new.monthly_listener_count < 0) THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'invalid monthly_listener_count';
    END IF;
END
```

5.2.5. Transaction_BEFORE_INSERT

```
CREATE TRIGGER `Transaction_BEFORE_INSERT` BEFORE INSERT ON `transaction`
 FOR EACH ROW BEGIN
        IF (new.amount> 0) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'invalid data';
    END IF;
END
```

5.2.6. User_BEFORE_INSERT

```
CREATE TRIGGER `User_BEFORE_INSERT` BEFORE INSERT ON `user`
 FOR EACH ROW BEGIN
        IF (NOT NEW.email LIKE ('%@%'))  THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'invalid email address...';
    END IF;
END
```

5.2.7.User_BEFORE_UPDATE

```
CREATE TRIGGER `User_BEFORE_UPDATE` BEFORE UPDATE ON `user`
 FOR EACH ROW BEGIN
        IF (new.credit < 0)  THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'invalid credit amount...';
    END IF;
END
```

5.3. Stored Procedures

5.3.1. Get Average Rating of a Playlist

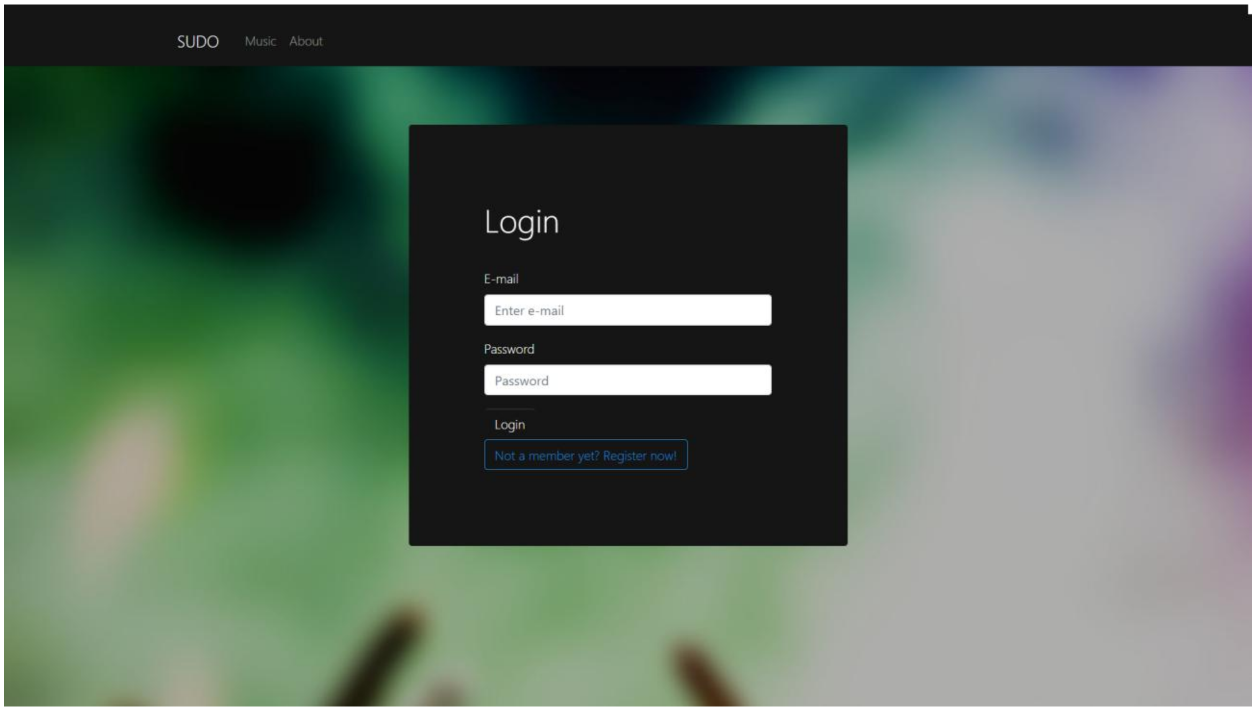
```
CREATE PROCEDURE `getAvgRating`(IN `ID` INT) NOT DETERMINISTIC CONTAINS SQL SQL
SECURITY DEFINER
BEGIN
        SELECT avg(rate_amount) FROM rates WHERE rates.playList_ID = ID;
END
```

5.3.2. Get Listener Count of an Artist

```
CREATE PROCEDURE `getListenerCount`(IN `artist_name` INT) NOT DETERMINISTIC
CONTAINS SQL SQL SECURITY DEFINER
BEGIN
SELECT listener_count FROM artist WHERE artist.artist_name = artist=name;
END
```

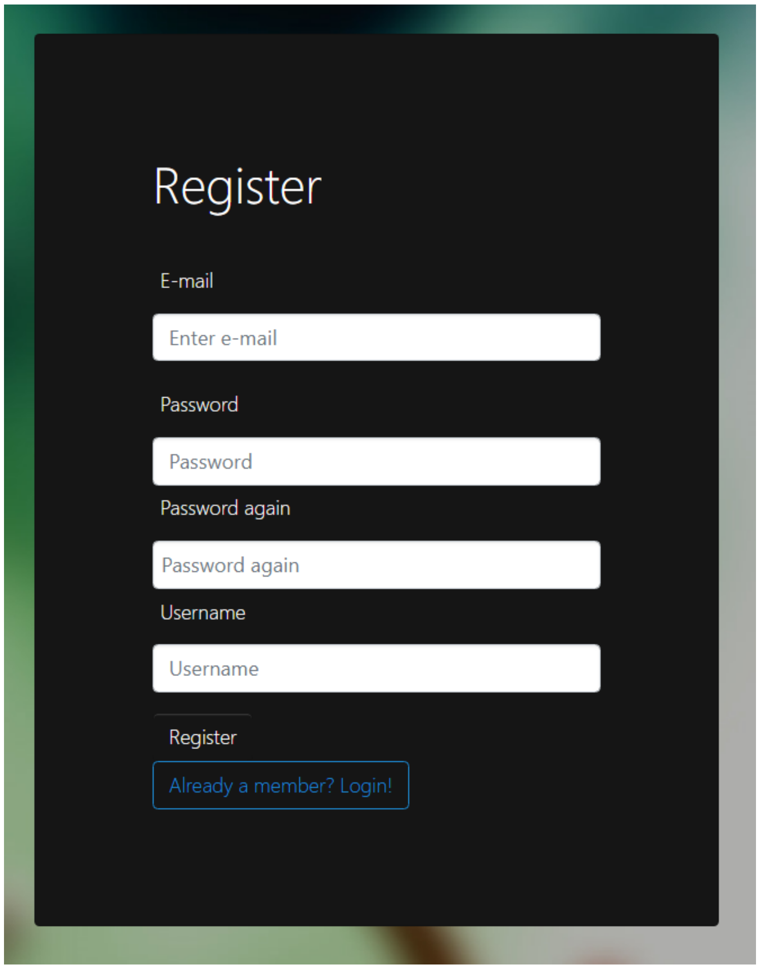
6. User Manual

6.1. Login Screen



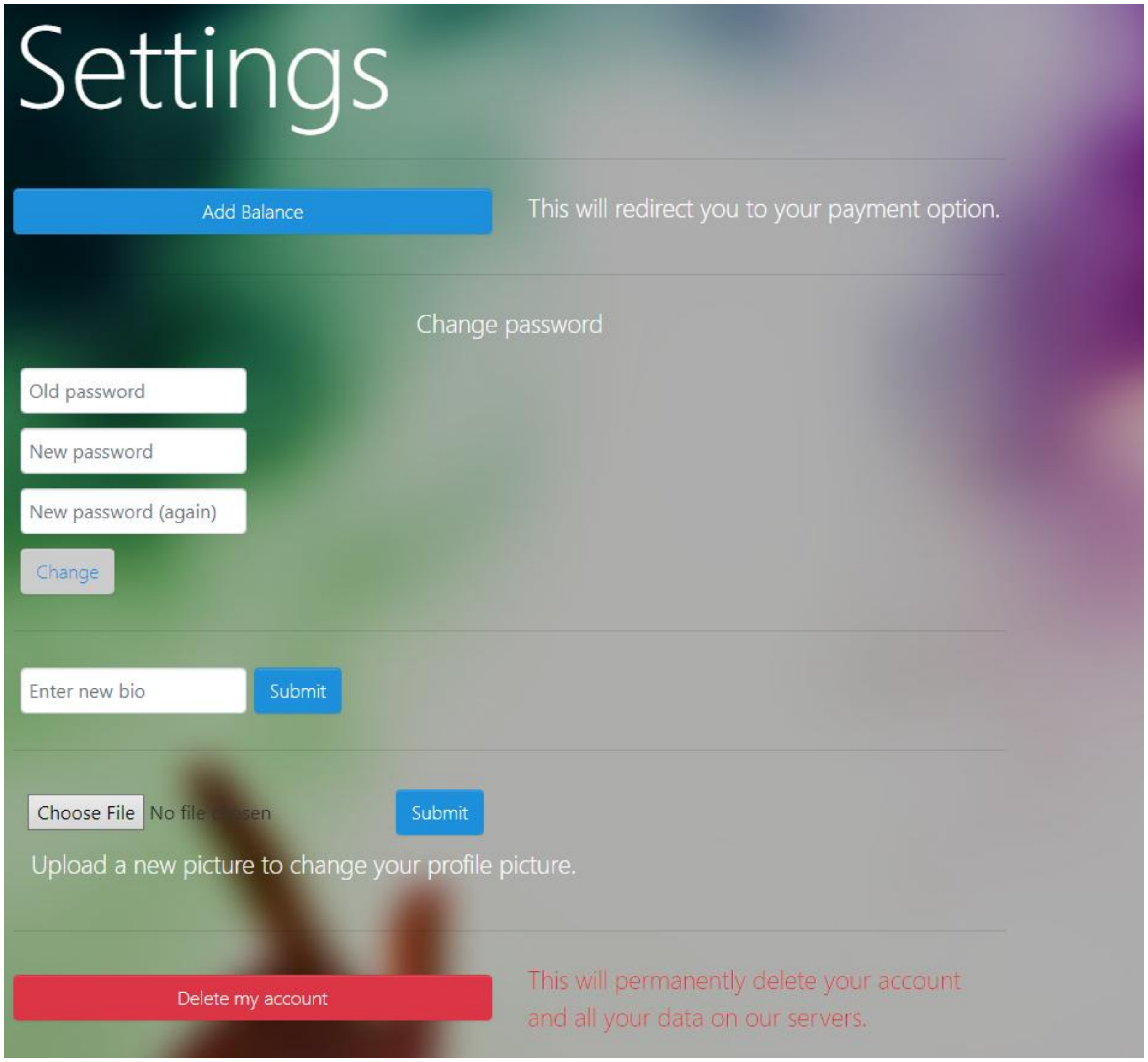
Users can login to their accounts by entering their usernames and passwords to the respective fields in the login screen. If a field is left blank or the user enters incorrect account information, the system displays error messages accordingly. If the account information is correct, then the user is directed to their profile page.

6.2. Register Screen



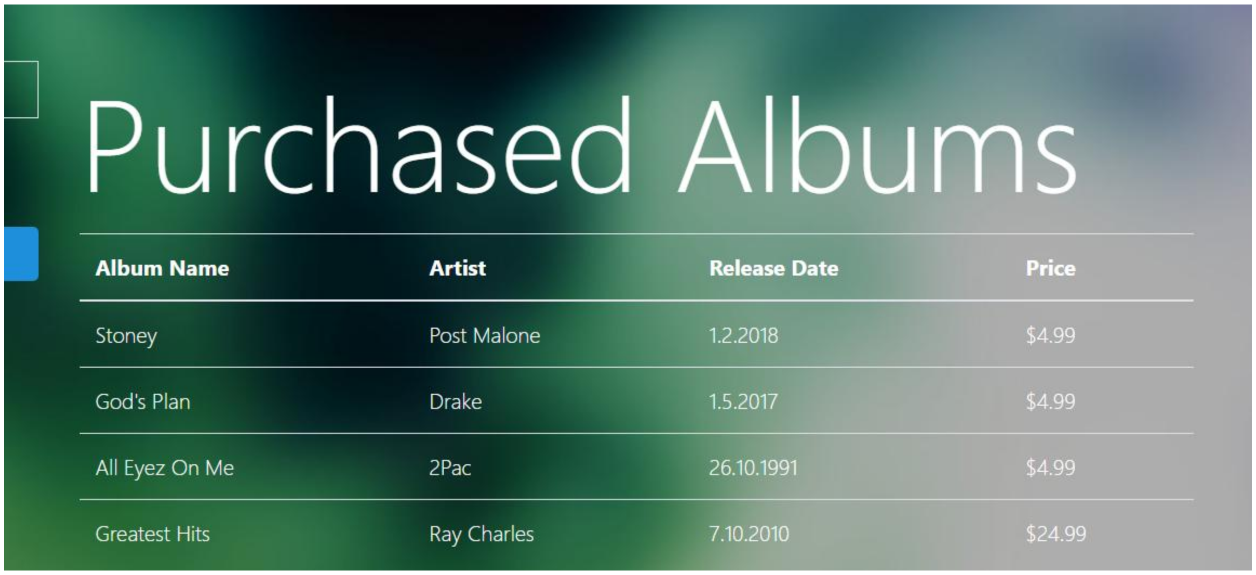
Users who are not yet a member of the website can register to our database using this UI element. It requires a valid e-mail address, a username and a password in order to complete the registration process.

6.3. Settings Screen


Settings
Add Balance
This will redirect you to your payment option.
Change password
Old password
New password
New password (again)
Change
Enter new bio
Submit
Choose File
No file chosen
Submit
Upload a new picture to change your profile picture.
Delete my account
This will permanently delete your account
and all your data on our servers.

In this page, users are able to change five different aspects of their account. They can add balance, change their password, change their bio, upload a new profile picture and delete their account.

6.4. Purchased Albums Screen

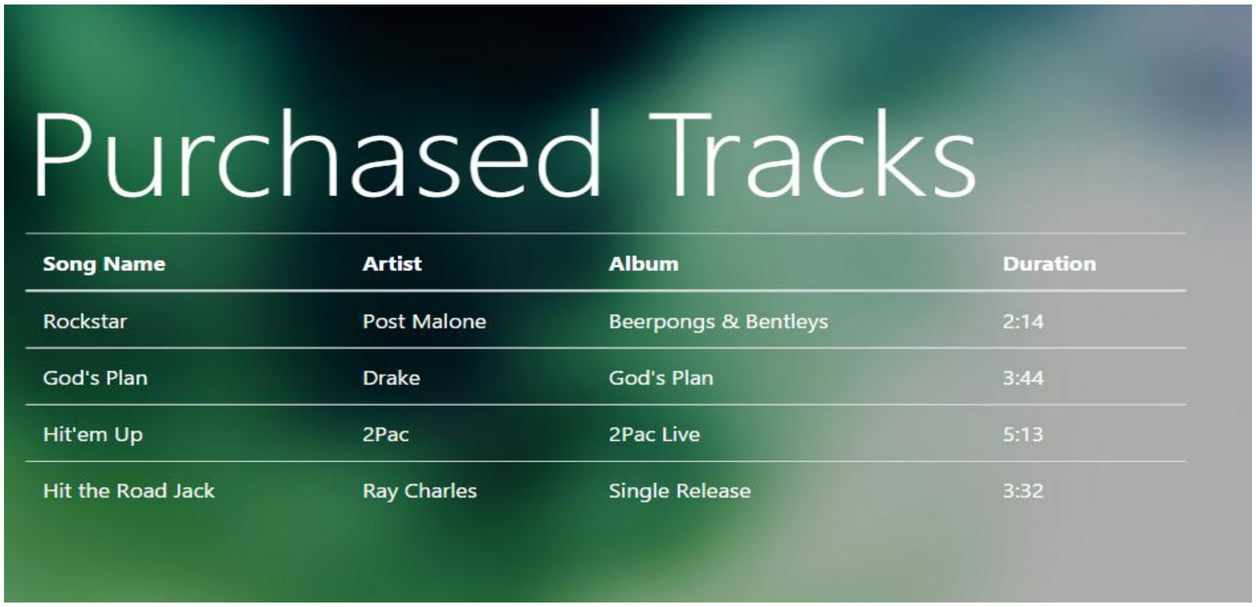


Purchased Albums

Album Name	Artist	Release Date	Price
Stoney	Post Malone	1.2.2018	$4.99
God's Plan	Drake	1.5.2017	$4.99
All Eyez On Me	2Pac	26.10.1991	$4.99
Greatest Hits	Ray Charles	7.10.2010	$24.99

In this page users are able to see the albums they have purchased. They can click on the album name in order to go that album’s page. The page also displays information regarding the artist, release dat and the price of the albums.

6.5. Purchased Tracks

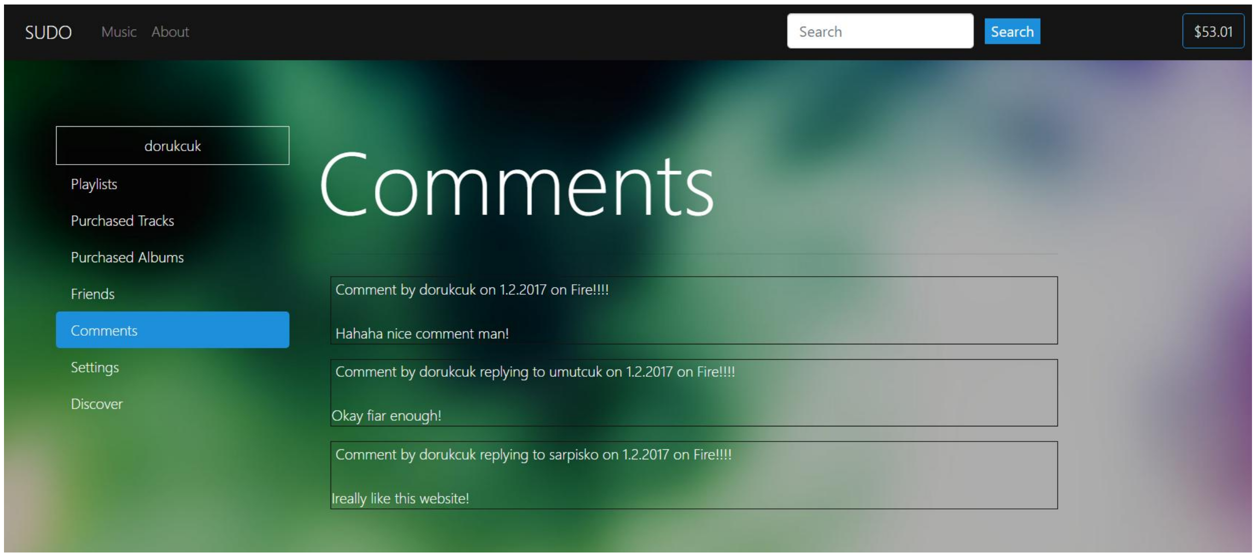


Purchased Tracks

Song Name	Artist	Album	Duration
Rockstar	Post Malone	Beerpongs & Bentleys	2:14
God's Plan	Drake	God's Plan	3:44
Hit'em Up	2Pac	2Pac Live	5:13
Hit the Road Jack	Ray Charles	Single Release	3:32

In this page users can view all of the tracks they have in their collection. This is a culmination of both all songs in the purchased albums and all songs released as singles. The page also displays additional information such as the artist of the track as well the album in which the songs was released and the duration of the track.

6.6 Comments Screen



Comments

Comment by dorukcuk on 1.2.2017 on Fire!!!!

Hahaha nice comment man!

Comment by dorukcuk replying to umutcuk on 1.2.2017 on Fire!!!!

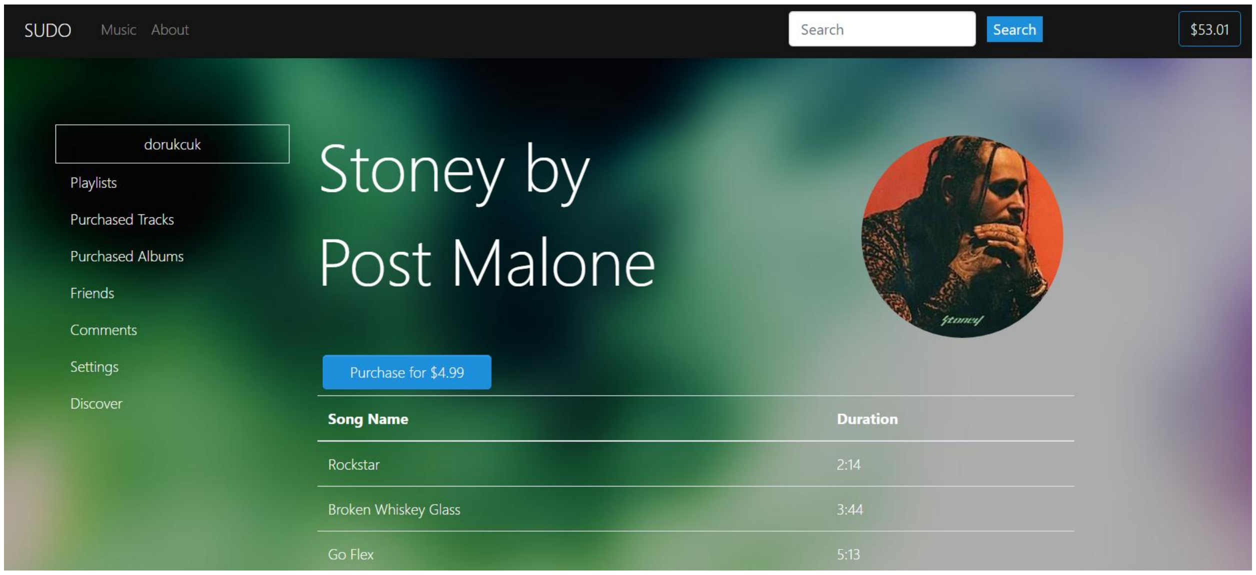
Okay fiar enough!

Comment by dorukcuk replying to sarpisko on 1.2.2017 on Fire!!!!

Ireally like this website!

Users can see the comments they made in the comments screen.

6.7 Display Album Screen

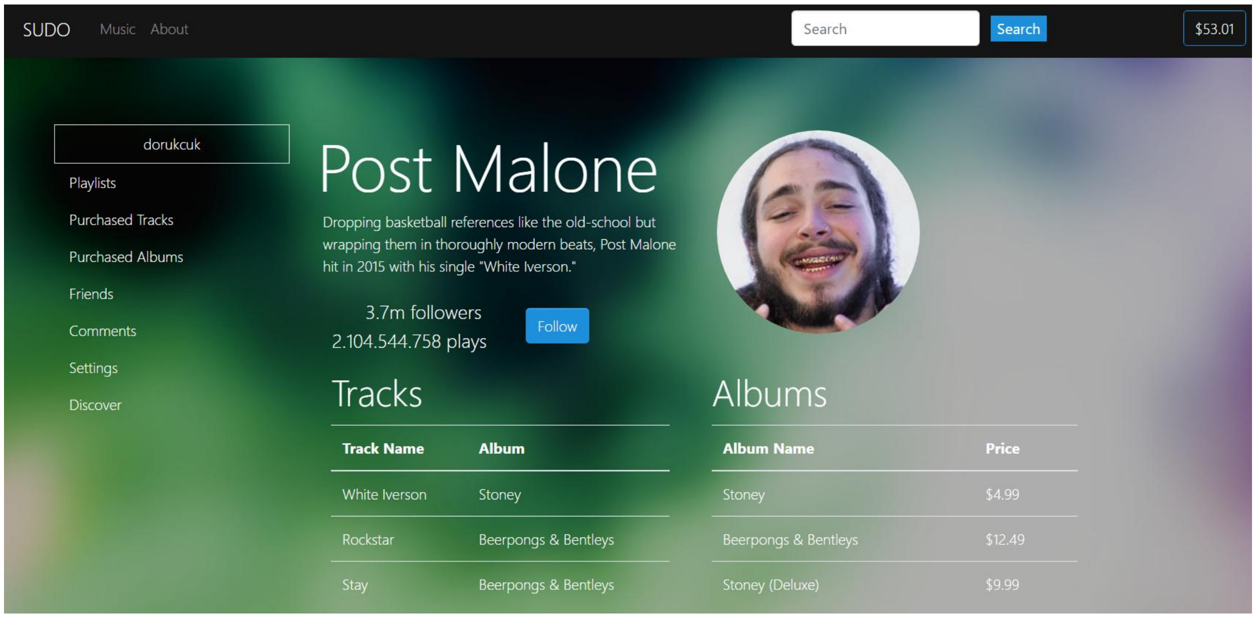


Stoney by Post Malone

Song Name	Duration
Rockstar	2:14
Broken Whiskey Glass	3:44
Go Flex	5:13

Users can display the albums of an artist and the songs inside that album in the isplay album screen.

6.8 Display Artist Screen



Post Malone

Dropping basketball references like the old-school but wrapping them in thoroughly modern beats, Post Malone hit in 2015 with his single "White Iverson."

3.7m followers
2.104.544.758 plays

Tracks

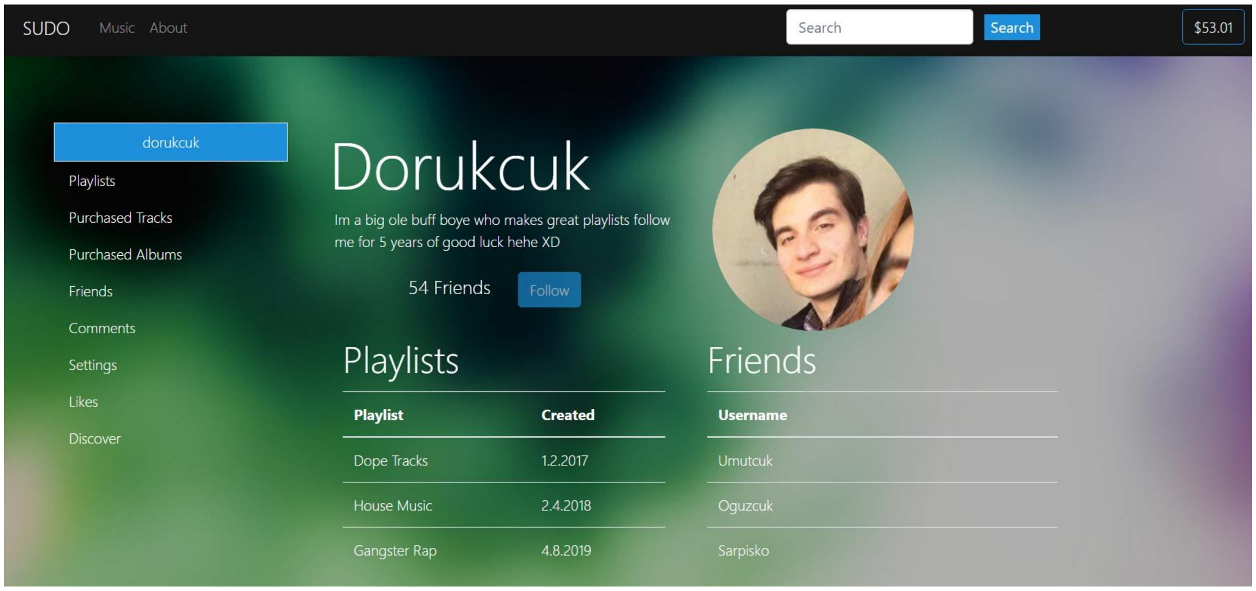
Track Name	Album
White Iverson	Stoney
Rockstar	Beerpongs & Bentleys
Stay	Beerpongs & Bentleys

Albums

Album Name	Price
Stoney	$4.99
Beerpongs & Bentleys	$12.49
Stoney (Deluxe)	$9.99

Users can display information about an artist in the Display Artist Screen. This screen includes the picture, bio, follower count, number of plays, tracks and albums of the artist. Users can also follow the artist in this screen.

6.9 Display User Screen



Dorukcuk

Im a big ole buff boye who makes great playlists follow me for 5 years of good luck hehe XD

54 Friends

Playlists

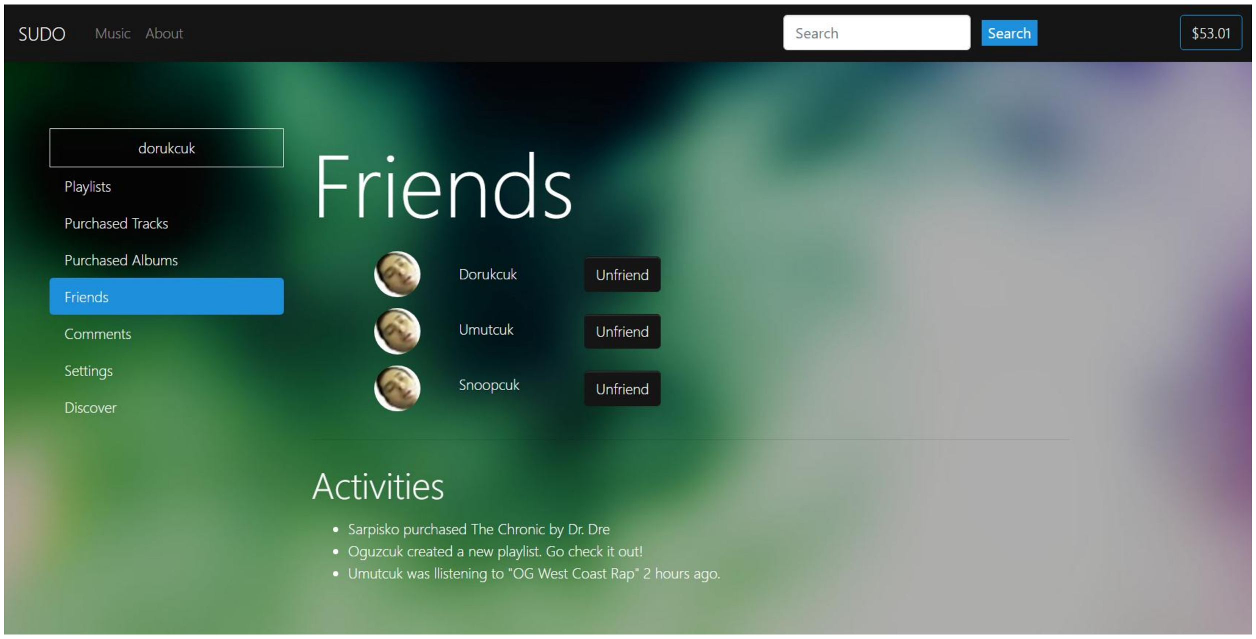
Playlist	Created
Dope Tracks	1.2.2017
House Music	2.4.2018
Gangster Rap	4.8.2019

Friends

Username
Umutcuk
Oguzcuk
Sarpisko

Users can display other users' profiles in display user screen. This screen displays bio, friend count, playlists and friends of the user.

6.10 Friends Screen

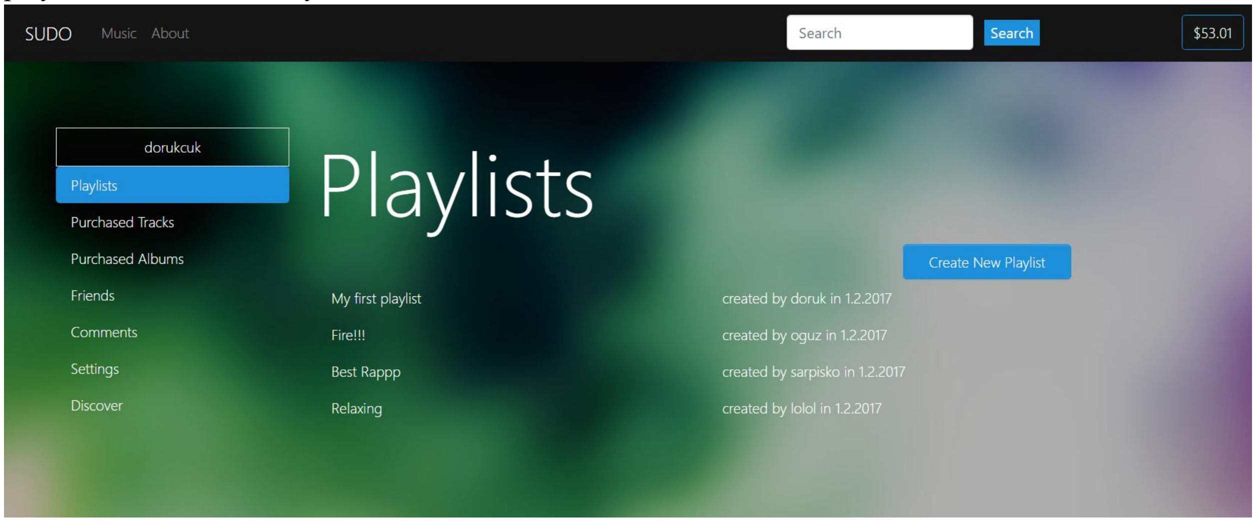


- Sarpisko purchased The Chronic by Dr. Dre
- Oguzcuk created a new playlist. Go check it out!
- Umutcuk was llistening to "OG West Coast Rap" 2 hours ago.

Users can see their friends and activity feed of their friends in the friends screen. Users can also click on unfriend buttons that are besides every friend name to remove them from their friends list.

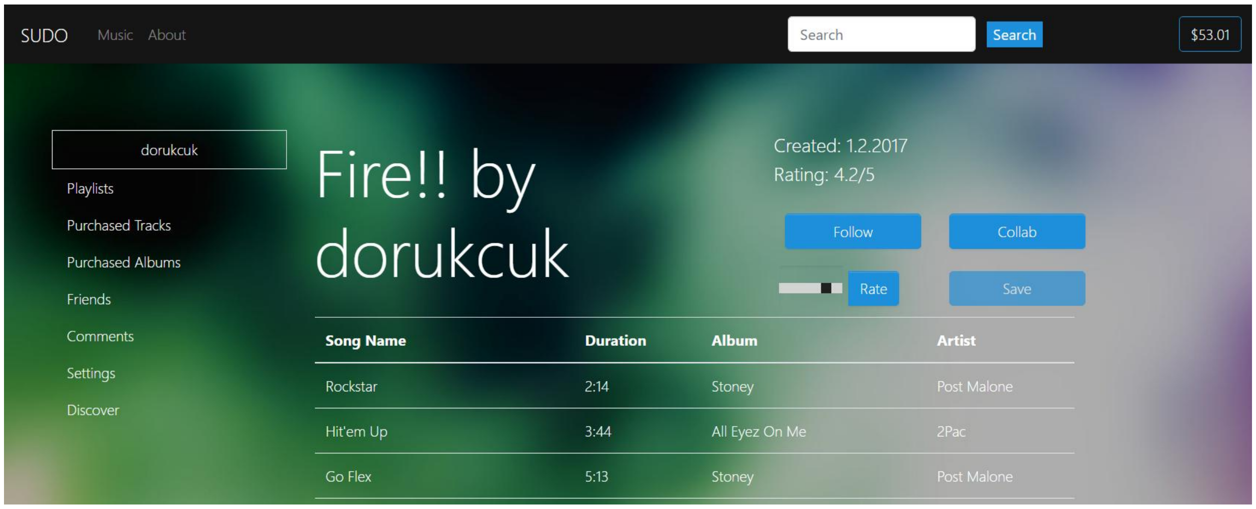
6.11 Playlists Screen

Users can display their saved playlists and create new playlists by clicking the “create new playlist” button in the Playlists screen.



My first playlist	created by doruk in 1.2.2017
Fire!!!	created by oguz in 1.2.2017
Best Rappp	created by sarpisko in 1.2.2017
Relaxing	created by lolol in 1.2.2017

6.12 Display Playlist Screen

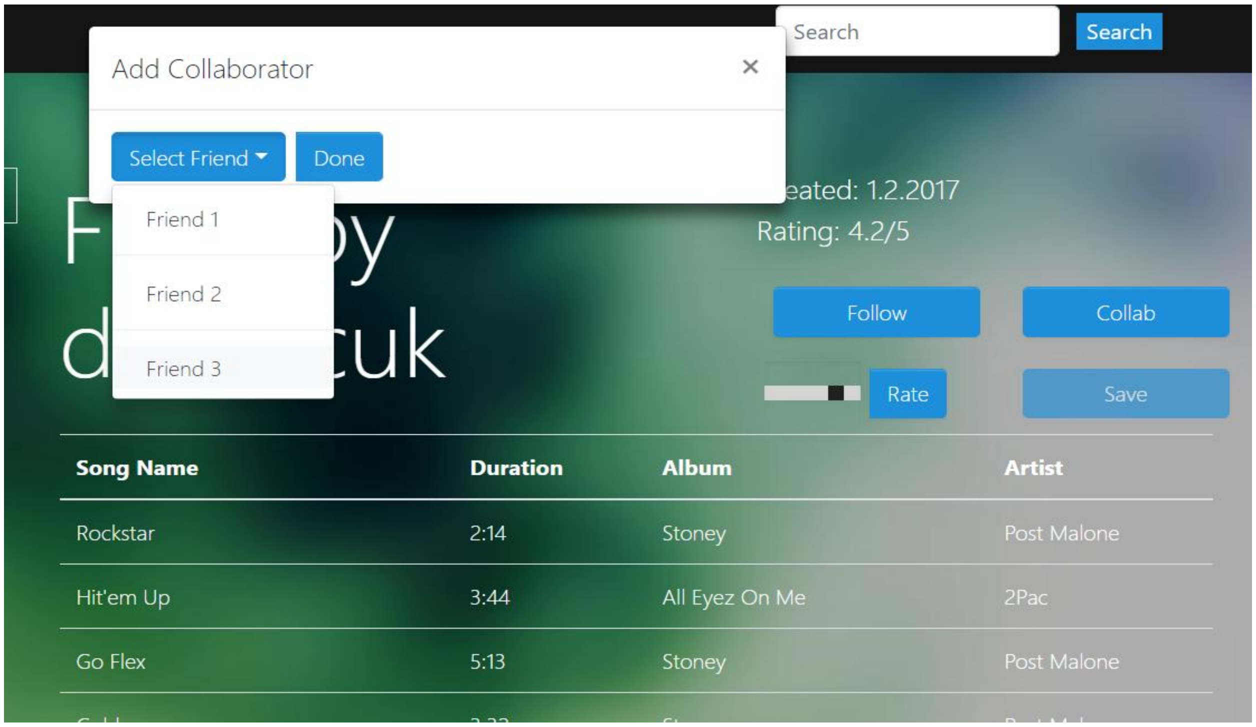


Fire!! by dorukcuk

Created: 1.2.2017
Rating: 4.2/5

Song Name	Duration	Album	Artist
Rockstar	2:14	Stoney	Post Malone
Hit'em Up	3:44	All Eyez On Me	2Pac
Go Flex	5:13	Stoney	Post Malone

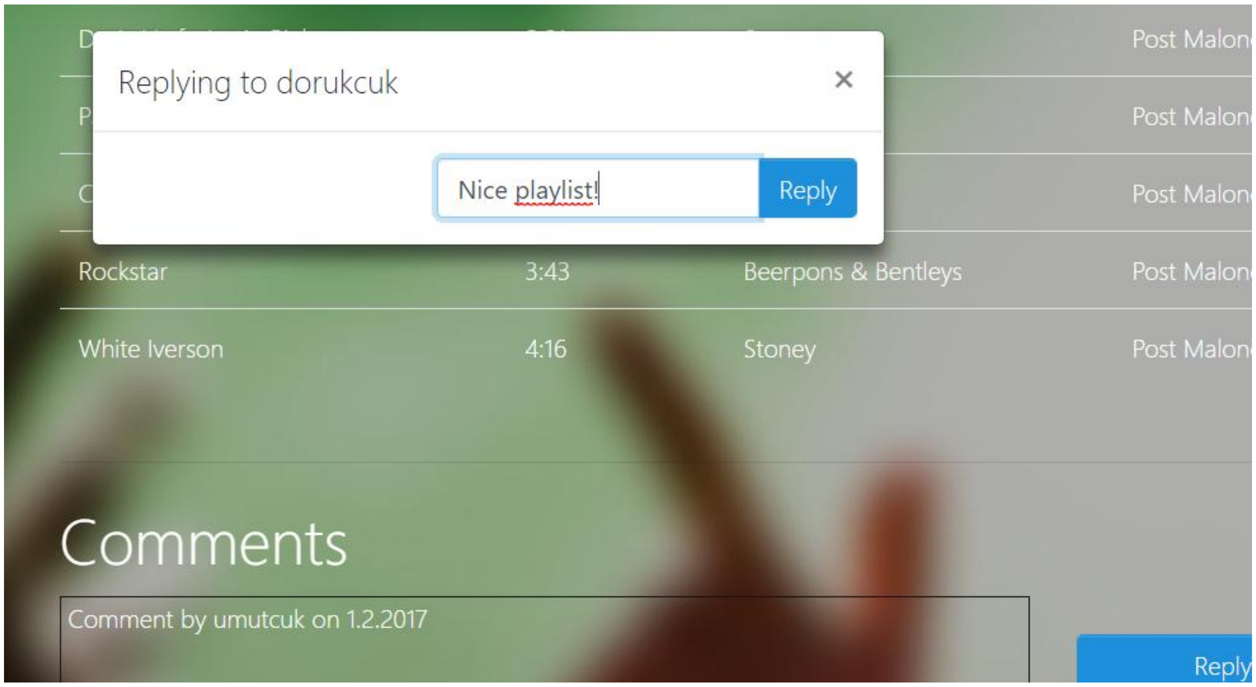
Users can display the contents of a playlist in the Display Playlist Screen. Users can also follow, add collaborators, rate and save the playlist in this screen. Users can also add comments and reply to them under the playlist.



eated: 1.2.2017
Rating: 4.2/5

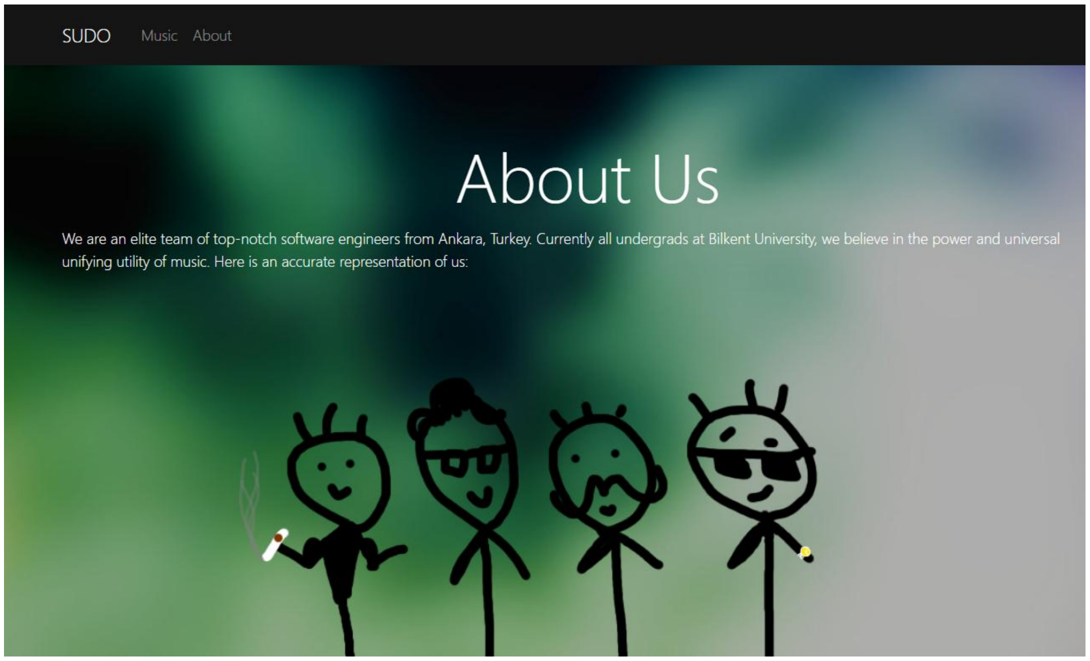
Song Name	Duration	Album	Artist
Rockstar	2:14	Stoney	Post Malone
Hit'em Up	3:44	All Eyez On Me	2Pac
Go Flex	5:13	Stoney	Post Malone

Users can add collaborators using the collab button which will display this popup menu (modal).


Replying to dorukcuk
Nice playlist!
Reply
Rockstar 3:43 Beerpons & Bentleys Post Malon
White Iverson 4:16 Stoney Post Malon
Comments
Comment by umutcuk on 1.2.2017
Reply

The reply button creates the reply popup (modal) where users can enter their comment and hit reply to save their comment.

6.13 About Screen

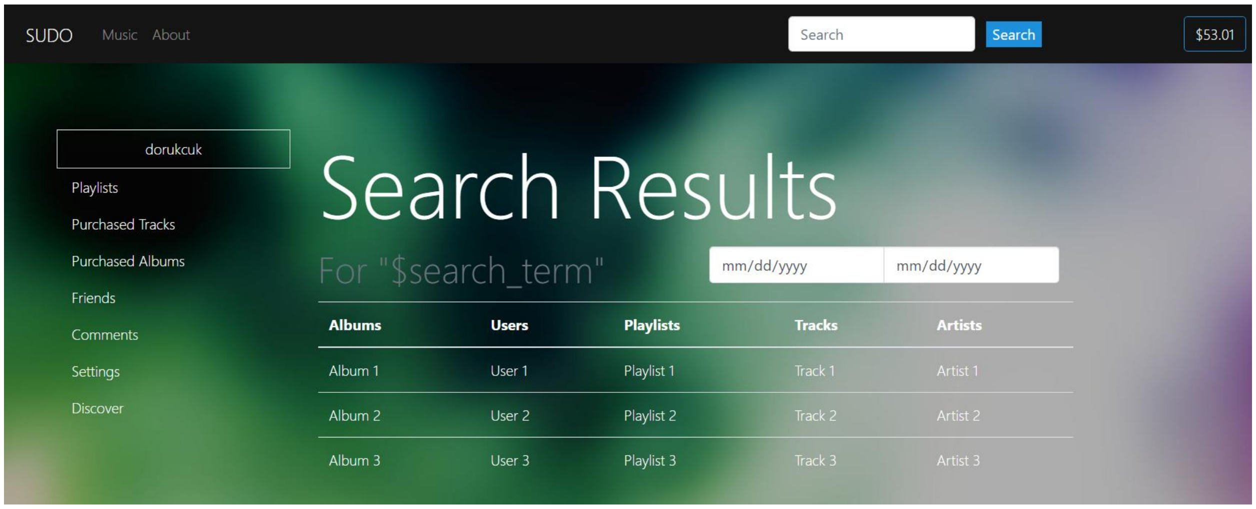


About Us

We are an elite team of top-notch software engineers from Ankara, Turkey. Currently all undergrads at Bilkent University, we believe in the power and universal unifying utility of music. Here is an accurate representation of us:

Users can display information about our team in the About screen.

6.14 Search Screen



Search Results

For "$search_term"

Albums	Users	Playlists	Tracks	Artists
Album 1	User 1	Playlist 1	Track 1	Artist 1
Album 2	User 2	Playlist 2	Track 2	Artist 2
Album 3	User 3	Playlist 3	Track 3	Artist 3

Users can display the search results of the terms they have searched using the search bar. This page includes albums, users, playlists, tracks and artists according to the text entered in the search bar. Users can also specify a range of dates so that only the results between that dates are displayed.

7. Website

Our project's source code and anything else regarding to the project will be hosted on a public GitHub repository. The webpage (at umutberkbilgic.github.io/Sudo-Music/) is hosted by GitHub using the GitHub Pages service.