SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

SRM Nagar, Kattankulathur– 603203

SCHOOL OF COMPUTING

DEPARTMENT OF NETWORKING AND COMMUNICATIONS

Course Project

Title: Multi4Connect – A FSD Project

Course Code **:** 18CSC310J

Course Name **:** Data Centric Networking and System Design

Faculty **:** Dr. S. Murugaanandam

Team Members:

1. Vijay Kumar Velpuri (RA2011028010040)
2. Uday Vamsidhar V Volety (RA2011028010008)
3. Harsh Pathak (RA2011028010022)
4. Osolla Ganapathi (RA2011028010045)

MULTI4CONNECT

The one stop for gaming, friends and music.

Abstraction

Over the last 2 decades, humans have made a lot of advancements in the field of technology. We had a shift from the industrial age to the age of data & electronics in just a matter of 25 years.
Everything today is an advancement or an improvement of its previous version. This is known as updates. However, one of the most popular fields & the most used field of the era has seen very little improvement & updates.
The field that is most loved, the field that allows the human neural circuit a real workout – games.
Yes, PC & mobile games have come a long way, and we are not arguing or even pointing fingers at these. Our focus is on browser-based games.
Since the start of the millennial, people have been more interested in mobile & PC applications.
However, with the 2021 news of Flash being supported by Google has ended, we have seen the value of browser-based games.
People still enjoy playing games online without having to commit storage or download anything.

Introduction

A browser game or a "flash game" is a video game that is played via the internet using a web browser. They are mostly free-to-play and can be single-player or multiplayer.

Some browser games are also available as mobile apps, PC games, or on consoles. For users, the advantage of the browser version is not having to install the game; the browser automatically downloads the necessary content from the game's website. However, the browser version may have fewer features or inferior graphics compared to the others, which are usually native apps.

The front end of a browser game is what runs in the user's browser. It is implemented with the standard web technologies of HTML, CSS, JavaScript, and Web Assembly. In addition, WebGL enables more sophisticated graphics. On the back end, numerous server technologies can be used.

In the past, many games were created with Adobe Flash, but they can no longer be played in the major browsers, such as Google Chrome, Safari, and Firefox due to Adobe Flash being shut down on December 31, 2020. Thousands of these games have been preserved by the Flashpoint project.

Problem?

Back in 1995 when the first browser-based game was made, everyone was hyped. They had a game to play without having to actually download it. People had something to enjoy.

Over the next couple of years, a bunch of other browser-based games started to come into play as well. Here is where our problem begins. With more competition, every game creator now has to also ensure that their game is best for people to play.

However, this was also, not the problem. Since the millennials, as phone & PC applications for games gave better graphics, better experience, and better gameplay, everyone switched to making these types of games.

This did not mean that browser-based games were dead but look at the graphics.

This was the first browser game ever made.

And this is now, is there really a 20 years' worth of improvement?

Drawbacks?

Tech innovations like AI in gaming are focused on creating smart – more interactive games for efficient user experiences.

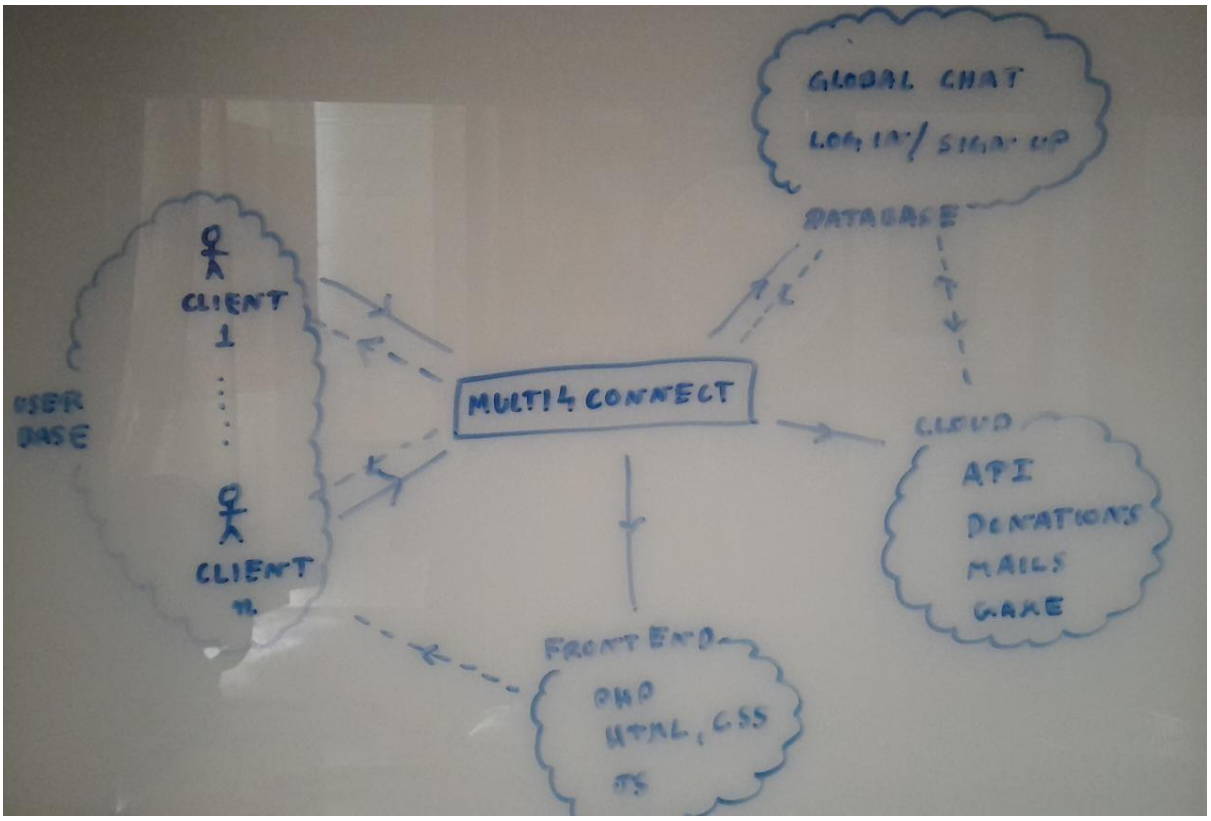
Talking in numbers, the annual revenue for tech gaming apps in 2021 was a 4.4% increase, making more than 180.3 billion USD.

The image below shows a clear segmentation of gaming revenues based on platforms and gaming categories for 2021.

What is being missed out on, are the following -

- Gesture Control.
- Facial Recognition.
- Voice recognition.
- Cutting-edge Graphics.
- High-definition Displays.
- NFT Game Development.
- Augmented Reality + Virtual Reality: Extended Reality.
- Offline Gaming Apps.

Approach & Flowchart




```
var firebaseConfig = {      apiKey:
"AIzaSyA_Cc4mrSvKui2XFg4puECgc9ejk2uVOcw",
    authDomain: "login-with-firebase-data-ddc55.firebaseapp.com",
projectId: "login-with-firebase-data-ddc55",
    storageBucket: "login-with-firebase-data-ddc55.appspot.com",
messagingSenderId: "1048671250983",
    appId: "1:1048671250983:web:25358d028c357a25b0fc02",
measurementId: "G-79J0Y6LXSF"
  };
  //Initialise variables   const auth
= firebase.auth()   const database =
firebase.database()


  //set up register function
function signup (){
//get all input fields
   username = document.getElementById('uname').value
password = document.getElementById('pass').value
confirmpassword = document.getElementById('pass').value
  }
```

```
  //validate input fields   if (validate_username(username) == false
|| validate_password(passwor d) == false){     alert('username or
Password is Otta Line!!')     return
  }


   //move on with auth
auth.createUserWithusernameAndPassword(username,password)
   .then(function(){     var
user = auth.currentUser
         var database_ref =
database.ref()
     var user_data = {
username : username,
password: password,
last_login : Date.now()
    }     database_ref.child('users/' +
user.uid).set(user_data)









    alert('User Created')


   })
   .catch(function(error){     var
error_code = error.code     var
error_message = error.message

alert(error_message)
   })

   function login() {     username =
document.getElementById('uname').value     password =
document.getElementById('pass').value


    if (validate_username(username) == false ||
validate_password(passw ord) == false){         alert('username or
Password is Otta Line!!')         return
      }
   }
auth.signInWIthusernameAndPassword(username,password)
```

```
   .then(function(){     var
user = auth.currentUser
         var database_ref =
database.ref()
     var user_data =
{
               last_login :
Date.now()
    }     database_ref.child('users/' +
user.uid).update(user_data)
     alert('User Logged
In')

   })
   .catch(function(error){     var
error_code = error.code     var
error_message = error.message

alert(error_message)
   })




  function validate_password(password){
if (password < 6){         return false
    }     else {
return true
    }
  }    function
validate_field(field) {
if(field == null){         return
false
    }     if (field.length
<=0){         return false
    }     else{
return true
    }
  }
```

Results & Conclusion

https://github.com/Uday-Vamsidhar/Chainlink-Multi4Connect

While our website offers a number of features for the scope of this report, we will show the API and the cloud we used for our database.

For the entire website video, please check out the video - https://www.youtube.com/watch?v=xq_B9hzQxMo The demo video is attached in the GCR as Mp4.

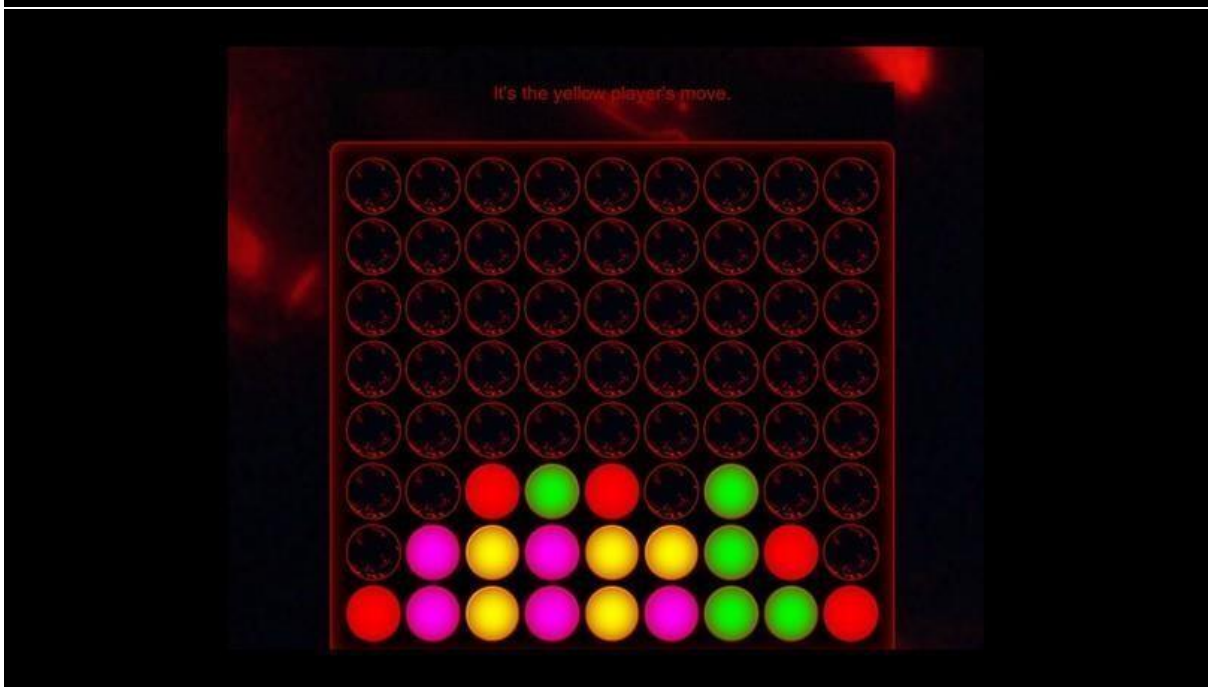
The team members are updated in the Excel sheet.

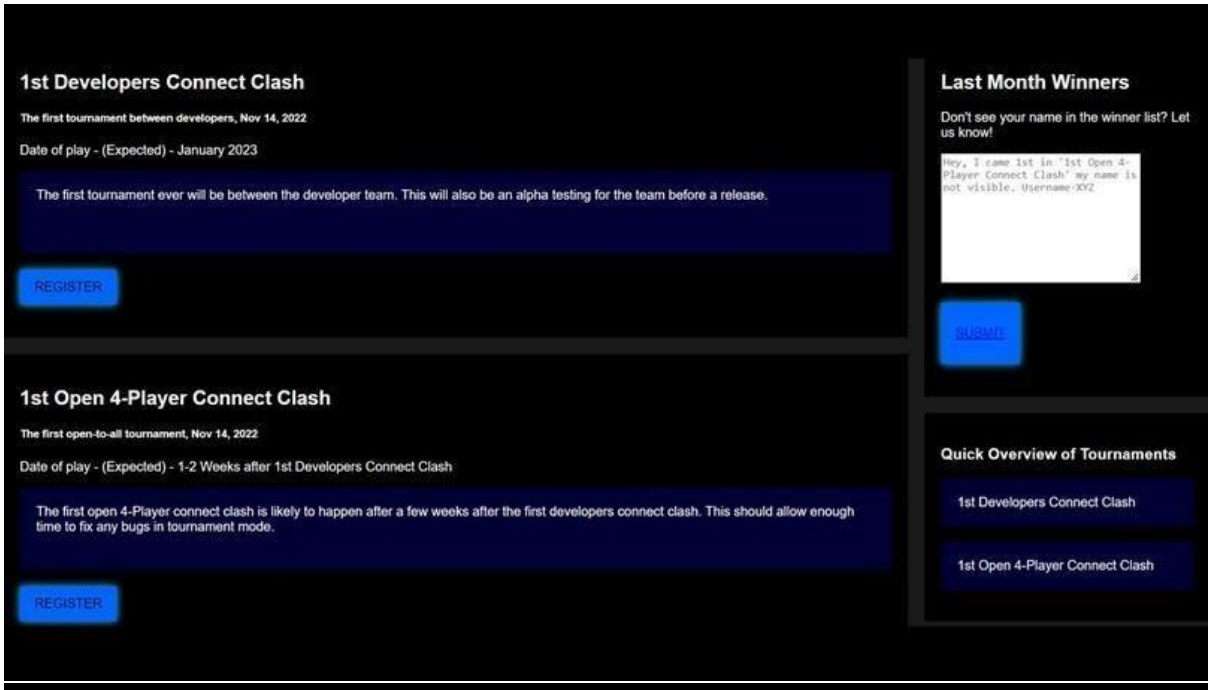
Here are some images of our work -

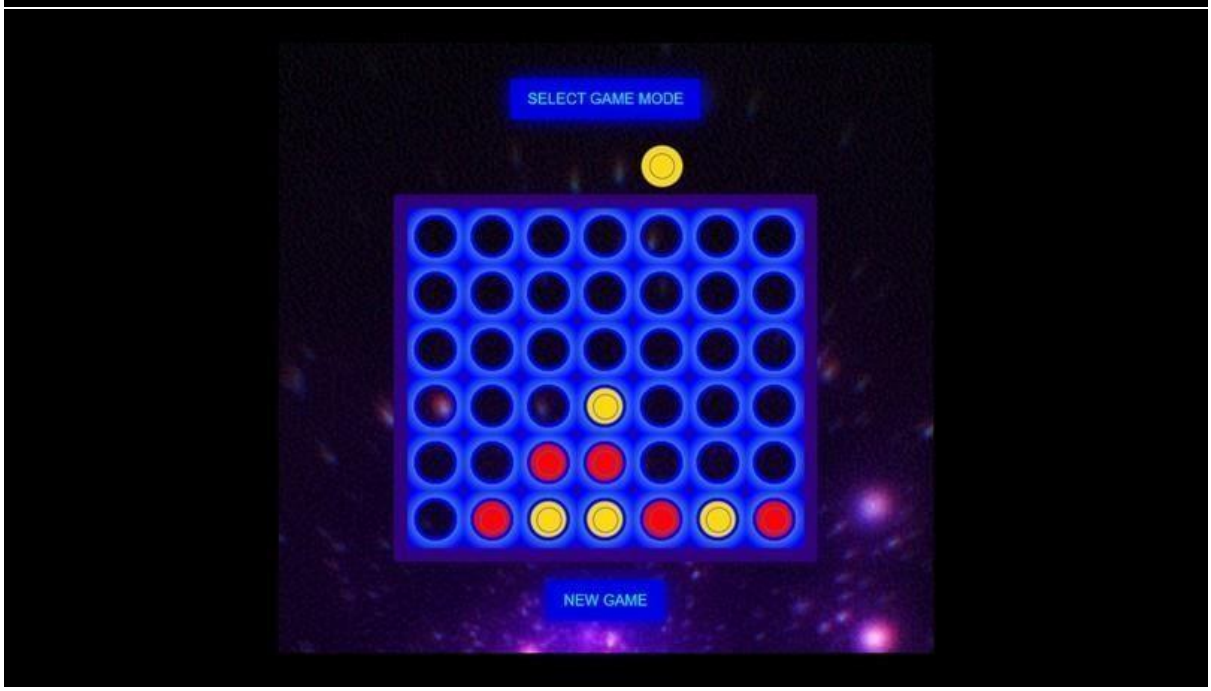

VS CPU
2 PLAYERS
4 PLAYERS
6 PLAYERS
8 PLAYERS


MULTI4CONNECT GLOBAL CHAT
Blank
Blank
Blank
Blank
Blank
Test
This is connected to the database
Test, type what you want here then hit the send button...
Test, Press this to change lobby name • Press Alt + Left Arrow got Home page


PLAY AGAINST THE CPU.
Presently, computer dominate humans in almost everything.
This bot will try to at least hold a draw.
Can you defend it? or better, call it a bluff?
This is the best way to improve your skills for competitive matches.
PLAY AGAINST CPU


It's the yellow player's move.


1st Developers Connect Clash
The first tournament between developers, Nov 14, 2022
Date of play - (Expected) - January 2023
The first tournament ever will be between the developer team. This will also be an alpha testing for the team before a release.
REGISTER
1st Open 4-Player Connect Clash
The first open-to-all tournament, Nov 14, 2022
Date of play - (Expected) - 1-2 Weeks after 1st Developers Connect Clash
The first open 4-Player connect clash is likely to happen after a few weeks after the first developers connect clash. This should allow enough time to fix any bugs in tournament mode.
REGISTER
Last Month Winners
Don't see your name in the winner list? Let us know!
Hey, I came 1st in '1st Open 4-Player Connect Clash' my name is not visible. Username:XYZ
SUBMIT
Quick Overview of Tournaments
1st Developers Connect Clash
1st Open 4-Player Connect Clash


SELECT GAME MODE
NEW GAME


0:00 / 3:10
GLOBAL CHAT
KNOW THE RULES
TOURNAMENTS
DONATE TO US
SELECT GAME MODE
SIGNUP/LOGIN
DEVELOPERS NOTE
LEADERBOARDS
UPDATES & BUGS
YOUR MUSIC


0:00 / 3:36
Multi4Connect
The one stop for Music

TASK 2

Objective:

The objective of this project is to generate music using AWS DeepComposer, which uses generative AI models like AutoRegressive Convolutional Neural Network (AR-CNN).

Methodology:

Autoregressive convolutional neural networks (AR-CNNs) are used to study systems that evolve over time and assume that the likelihood of some data depends only on what has happened in the past. It is a useful way of looking at many systems, from weather prediction to stock prediction.

The AR-CNN generative technique uses a U-Net architecture. It was originally developed for image generation, a type of computer vision task.

The available AR-CNN model in AWS DeepComposer was trained using a dataset that consists of chorales written and composed by Johann Sebastian Bach. To train the model, the audio files are first converted into piano roll images. Then, during model training, notes are first randomly added and removed from the piano roll images. Next, the model learns to detect them and replace the notes with new ones based on the dataset.

When you perform inference with the AR-CNN model in the AWS DeepComposer music studio, the model will first attempt to detect notes that sound missing or out place. The model determines if a note is missing or out of place by comparing it against dataset it was trained on. It then replaces those notes with ones that are likely to appear in the dataset it was trained on.

Using the different available model parameters, you can modify to what extent the model will add and/or remove notes during inference. The model parameters are:

Sampling iterations

Controls the number of times your input melody is passed through the model. Increasing the number of iterations results in more notes being added and removed from the melody.

Maximum input notes to remove

Controls the percentage of input melody to be removed during inference. By increasing this parameter, you are allowing the model to use less of the input melody as a reference during inference. After performing inference, you can use the Edit melody tool to further modify your melody.

Maximum number of notes to add

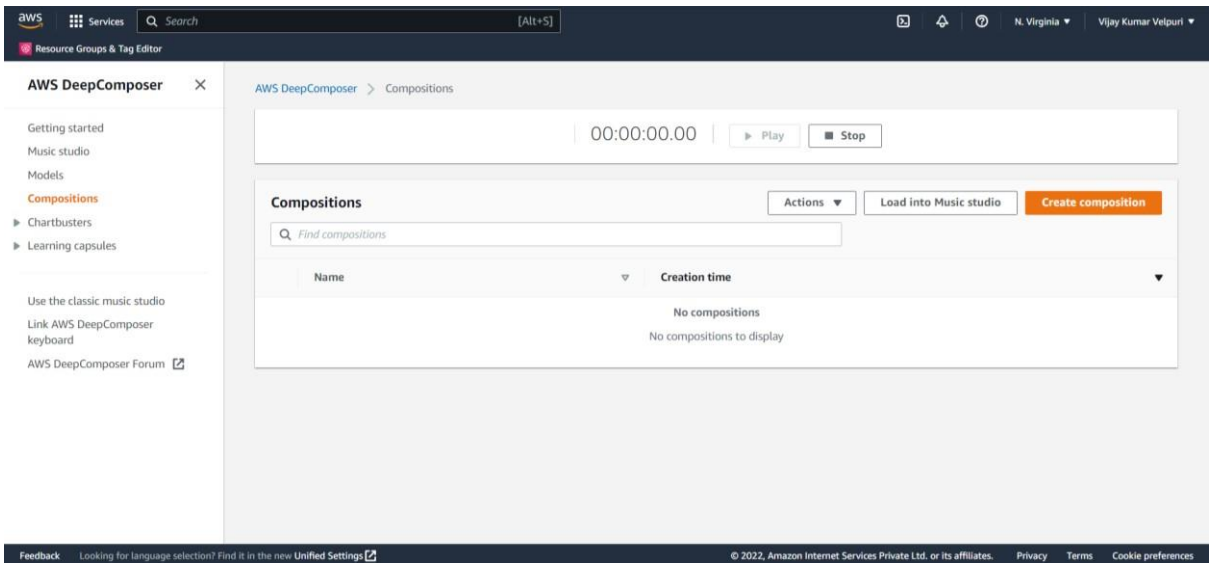
Controls the number of notes that can be added to the input melody. By increasing this number, you might introduce notes that sound out of place in your melody. It's also a creative way to experiment with your chosen melody. After performing inference, you can use the Edit melody tool to further modify your melody.

Creative risk

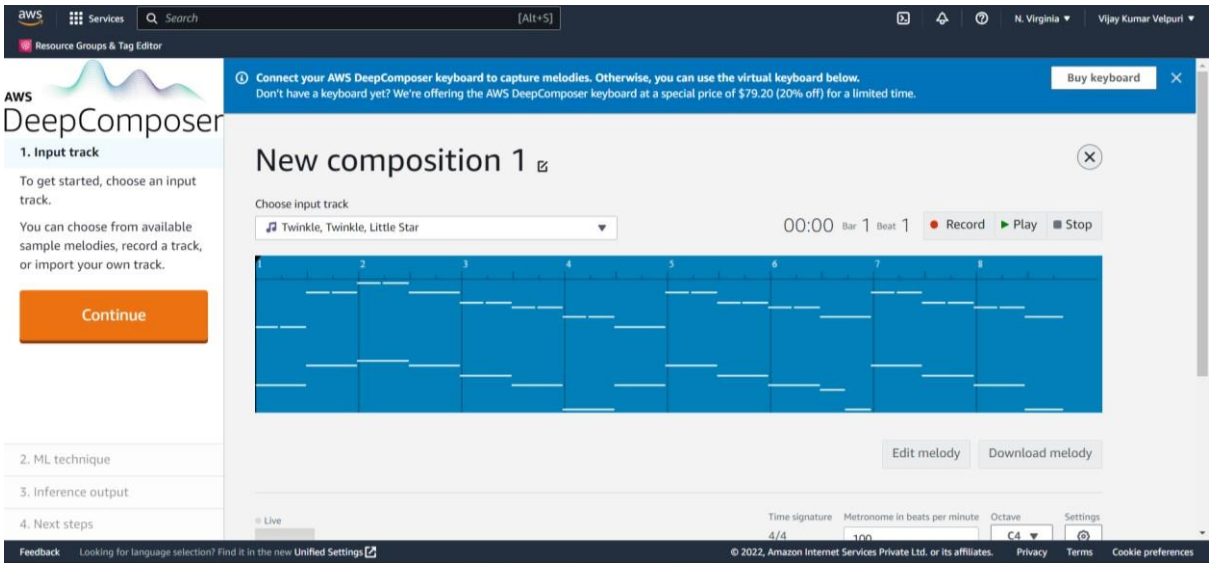
Controls how much the model can deviate from the music that it was trained on. More technically, when you change this value, you are changing the shape of the output probability distribution. If you set this value too low, the model will choose only high-probability notes. If you set this value too high, the model will more likely choose lower-probability notes.

Implementation:

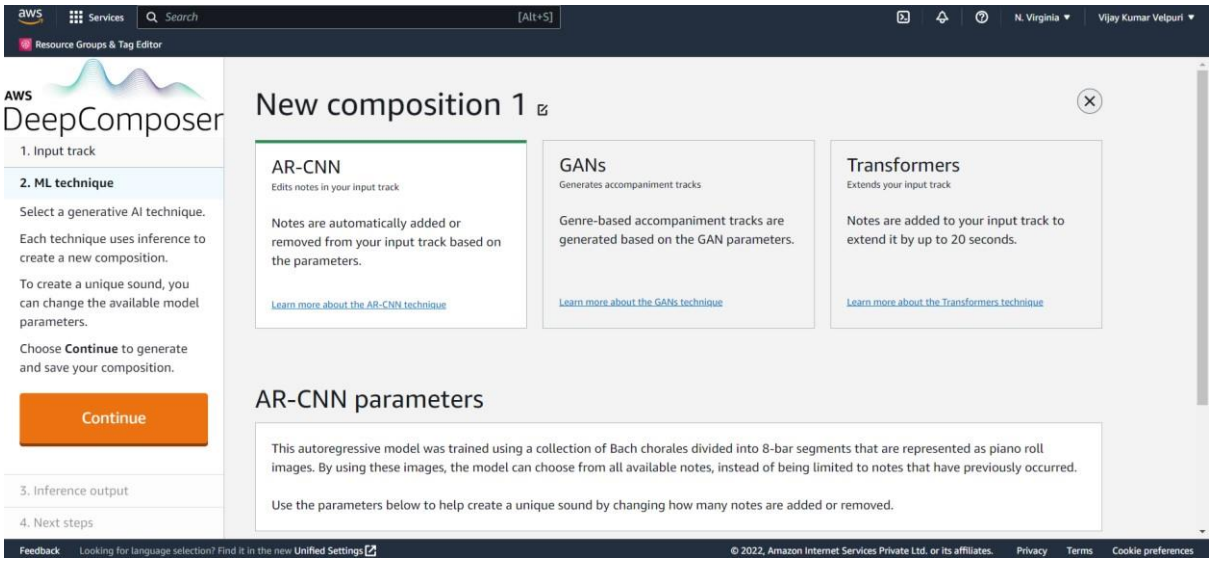
1. Opening AWS DeepComposer and going to the Compositions section.



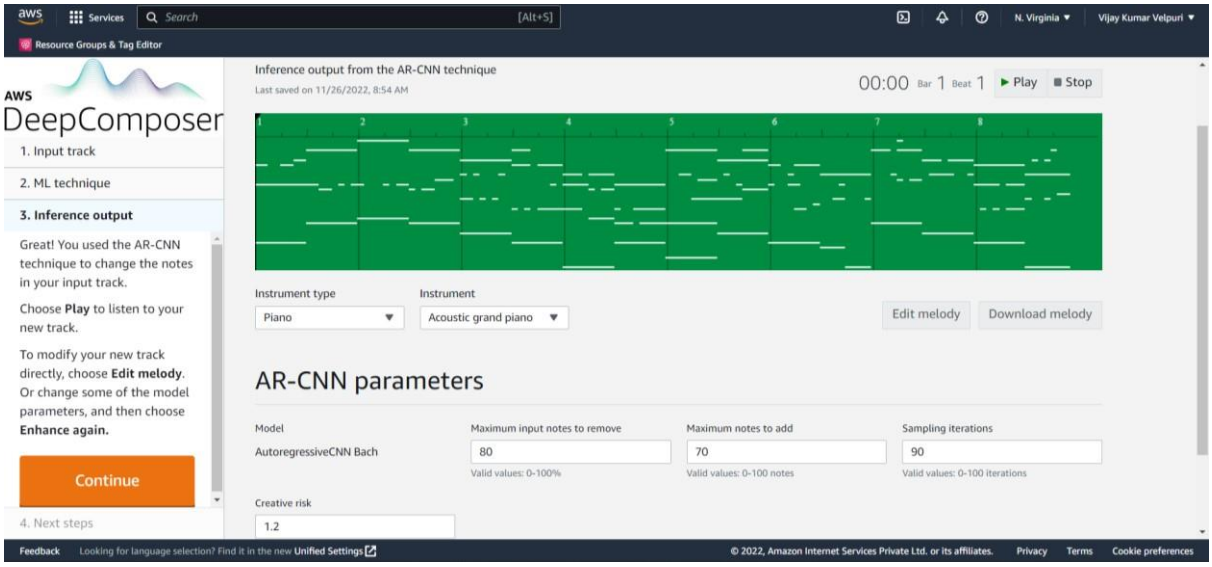
2. Choosing an input track. “Twinkle Twinkle Little Star” is chosen from the available presets.



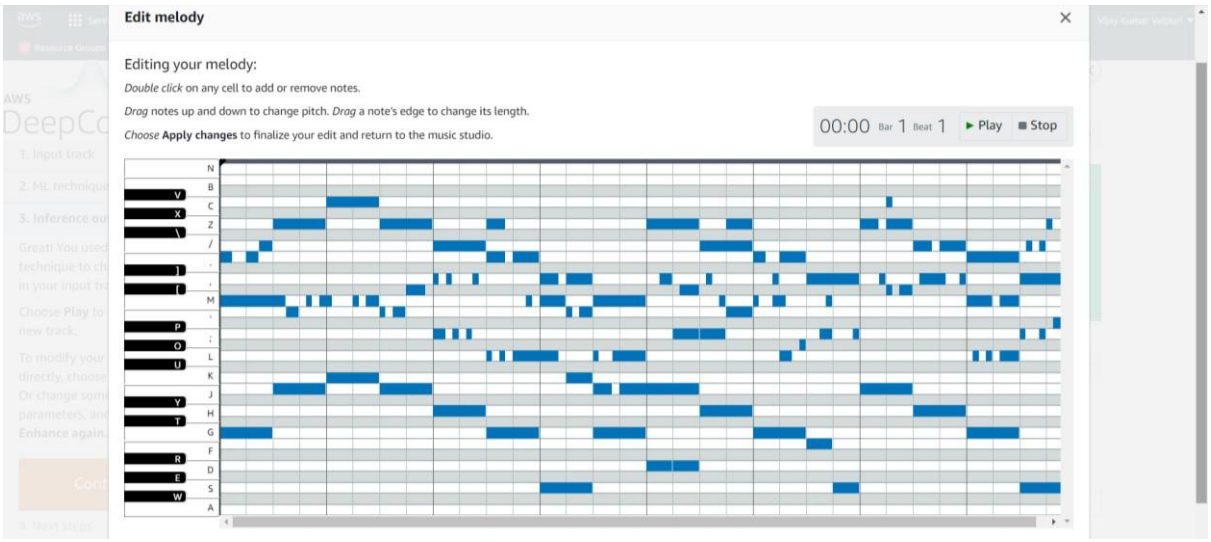
3. Choosing the ML technique to apply to modify the track and changing the parameters. AR-CNN is chosen as the ML technique. Configuring the AR-CNN parameters as follows:
    a. Maximum input notes to remove- 80
    b. Maximum notes to add- 70
    c. Sampling iterations- 90
    d. Creative risk- 1.2



4. Inference output from the AR-CNN technique.



5. Using Edit melody to apply some manual changes to the track.



6. Downloading the generated track by clicking on Download melody.

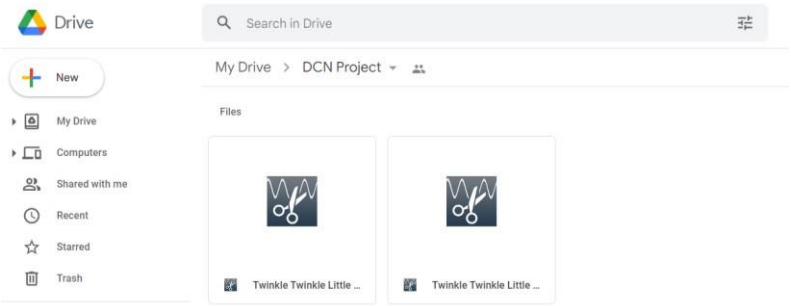
Output:

Music has been successfully generated using AWS DeepComposer.

Proof:

Attached herewith is the drive link containing the input track and the track generated by AWS DeepComposer in MIDI format.

https://drive.google.com/drive/folders/1arjS32dD3XU6tEwb7jIsHHh0qjIarcBV?usp=share_link



References:

https://docs.aws.amazon.com/deepcomposer/latest/devguide/gen-tech-arcnn.html