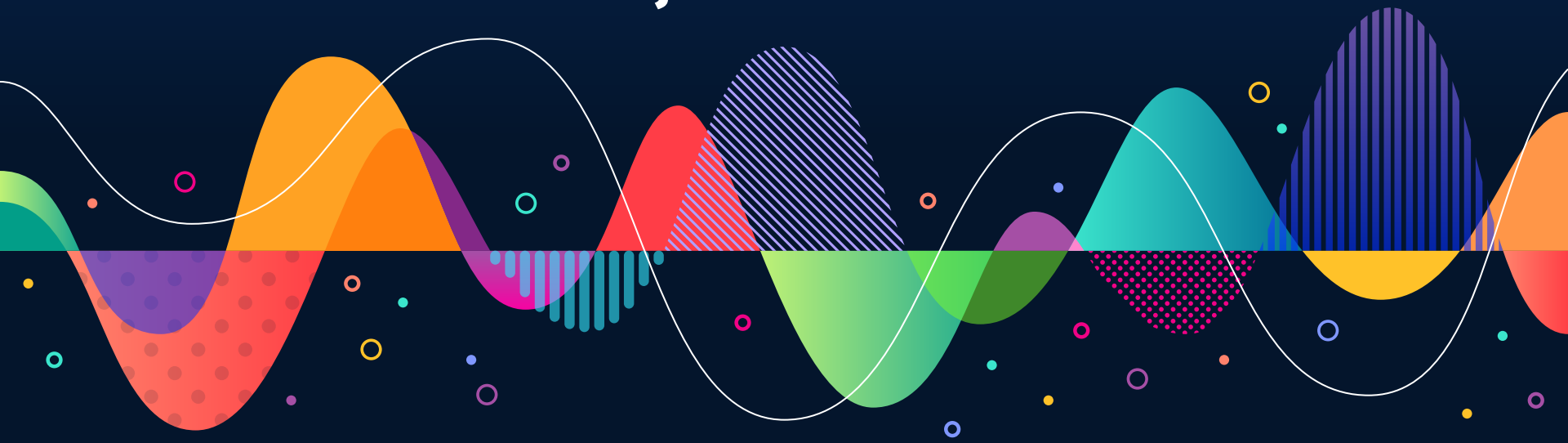
TWINKLE TINES

Final Project Presentation



Introduction

Hardware Team

- Stuart Pearson (EE) - Note Detection
- Eileen Hillier (EE) - Device Mount Design and Implementation
- Isaac Vrba (CPR E) - Microcontroller & LED Implementation

Introduction

Software Team

- Kaitlyn Nolting (SE) - Bluetooth
- Andrew Adams (SE) - Database Querying
- Mesa Hassel (SE) - Database Setup and Querying, UI Implementation
- Daniel Duerr (SE) - Future Plans

Project Inspiration - Stuart

Electrical Engineering is Hard

Reaching for my phone to relax/reset is not ideal

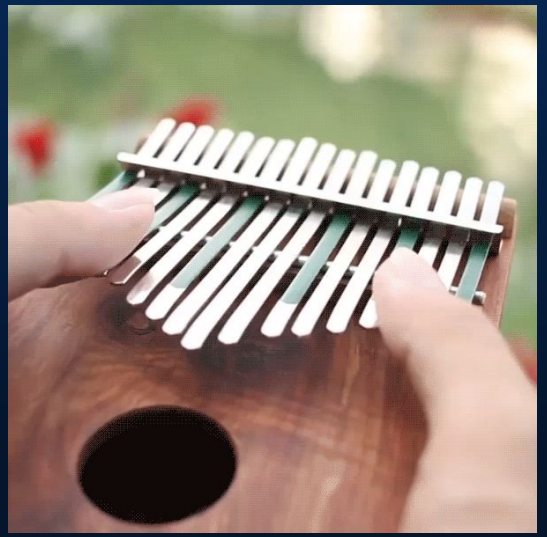
Project Inspiration - Stuart

Christmas Time 2021

Project Inspiration - Stuart

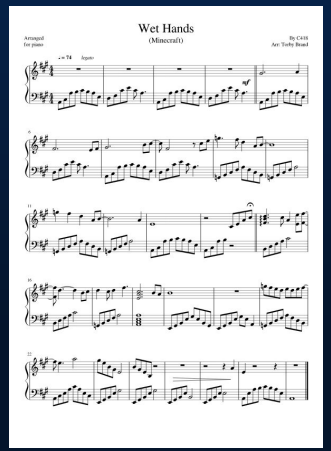
A Kalimba!

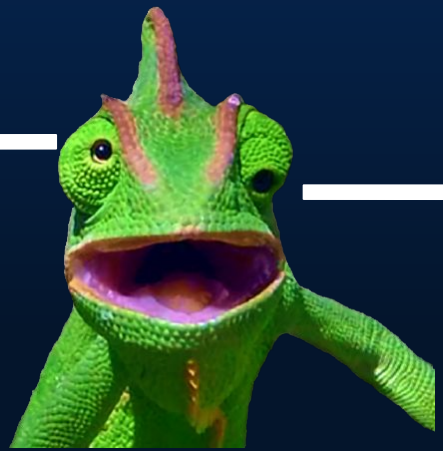
Notice the thumb dexterity here

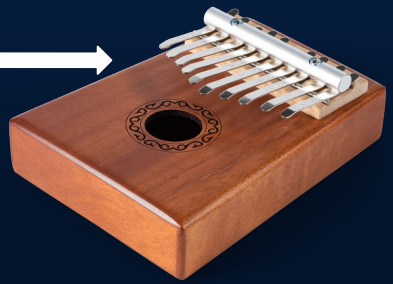


Project Inspiration - Stuart

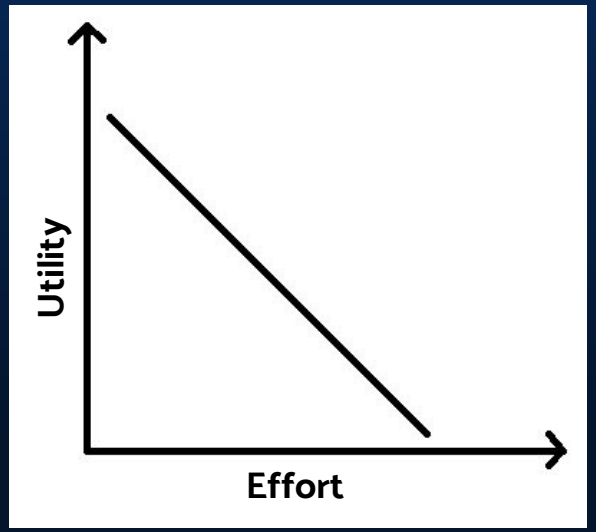
Simultaneously reading music and playing is very hard







Project Inspiration - Stuart



The death of my desire

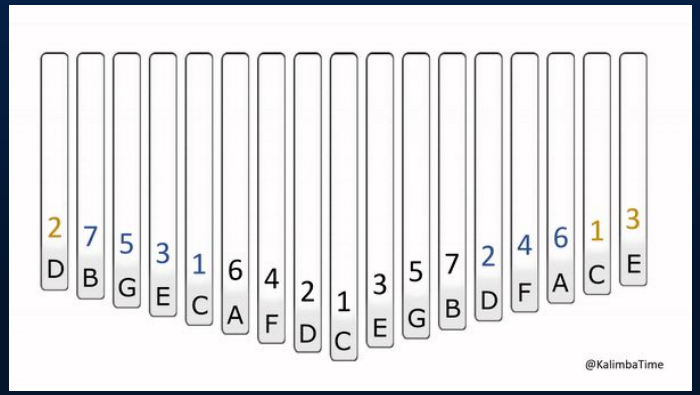
Memorizing music is too much effort to relax/reset/destress

Project Inspiration

Unless...

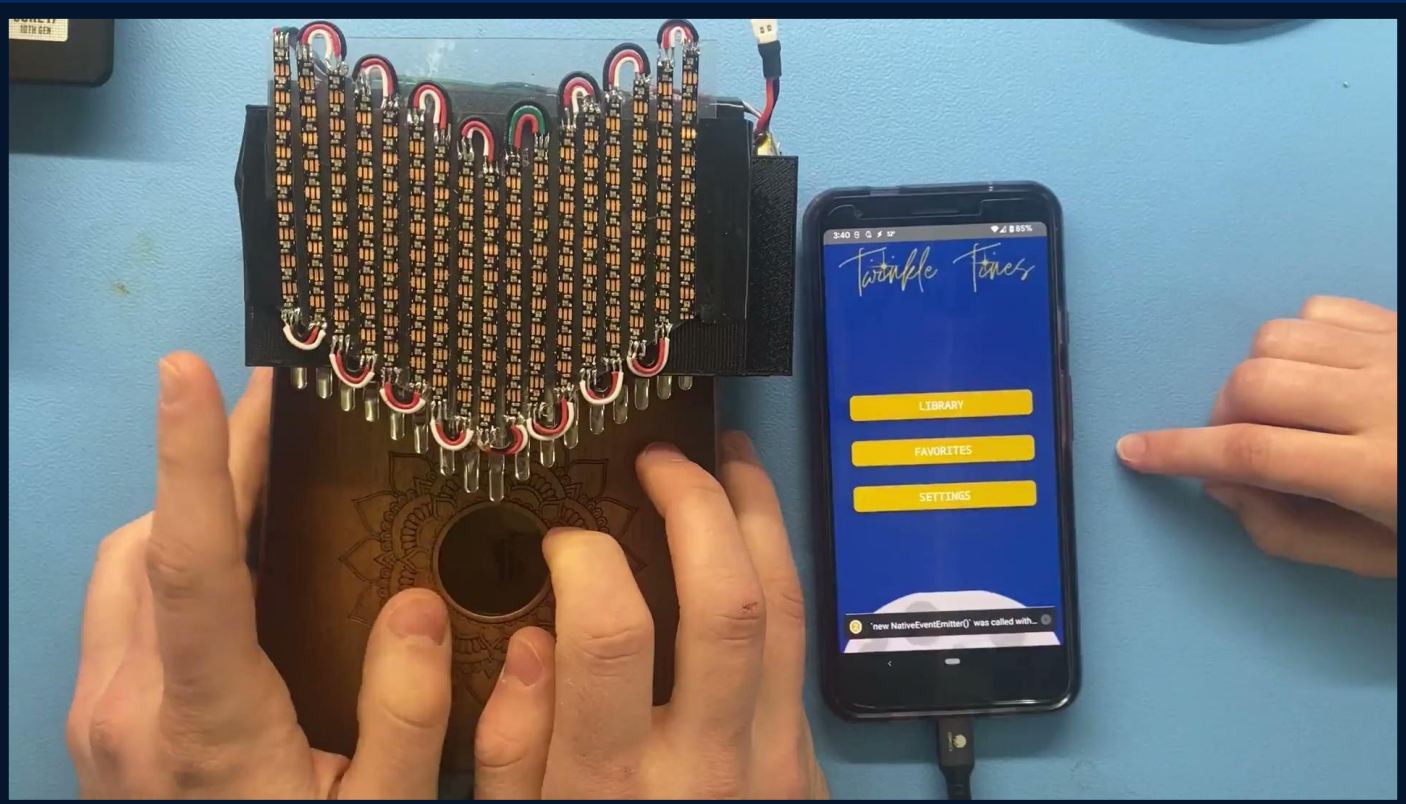
Project Inspiration - Stuart

Light up tines? A software interface?



The idea for Twinkle Tines was born!

Demo Video


Twinkle Tunes
LIBRARY
FAVORITES
SETTINGS
'new NativeEventEmitter()' was called with...

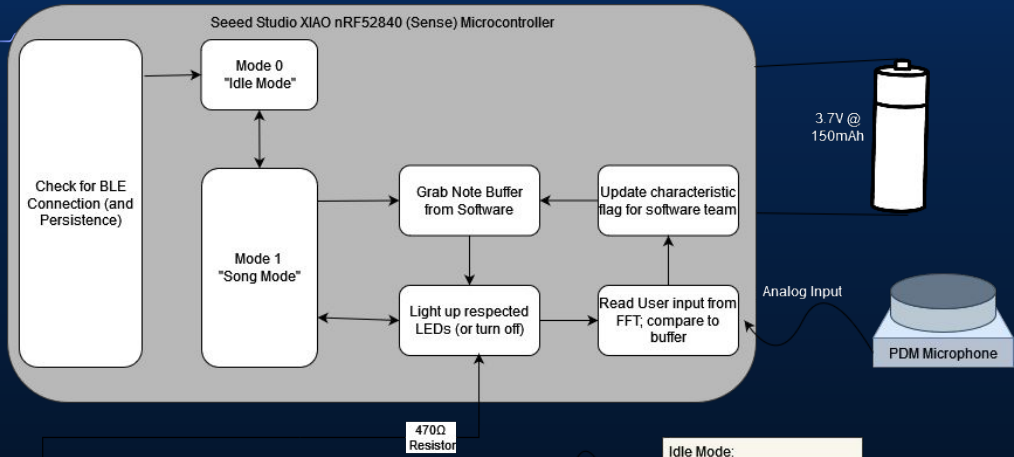
Implementation Architecture: Hardware - Isaac

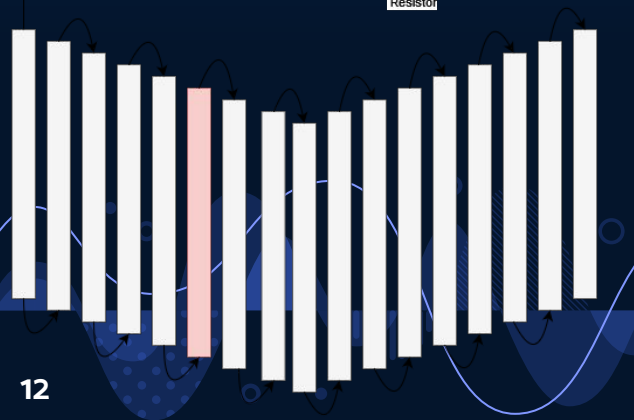
Hardware Used in Final Design

- Seeed Studio XIAO nRF52840 (Sense)
  - Built-in PWM and BLE module
- NeoPixel 1515 LED Strip 4mm wide (x2)
- 3.7V 250mAh Battery
- 470Ω Resistor

Microcontroller code

- Utilizes Bluetooth Low Energy (BLE) for connection and persistence to application
- AdafruitLED Library used for interacting with the LEDs
- PDM analog input passes through a FFT to dissect notes played
  - Comparison between user input and actual note that is to be played





Idle Mode:
- None of the LEDs are lit up.
- System waiting on user to enter song mode.

Song Mode:
- Entire tine lights up for the next note tine that is to be played. Tine says lit up until user presses the correct tine.

Future Waterfall Mode:
- The LEDs will produce a waterfall effect of the incoming notes over time.
- Tines are to be pressed when the bottom-most LED is lit up (~.3s window)

NeoPixel 1515 LED Strip 4mm wide (17 strips of 10 LEDs connected in series)

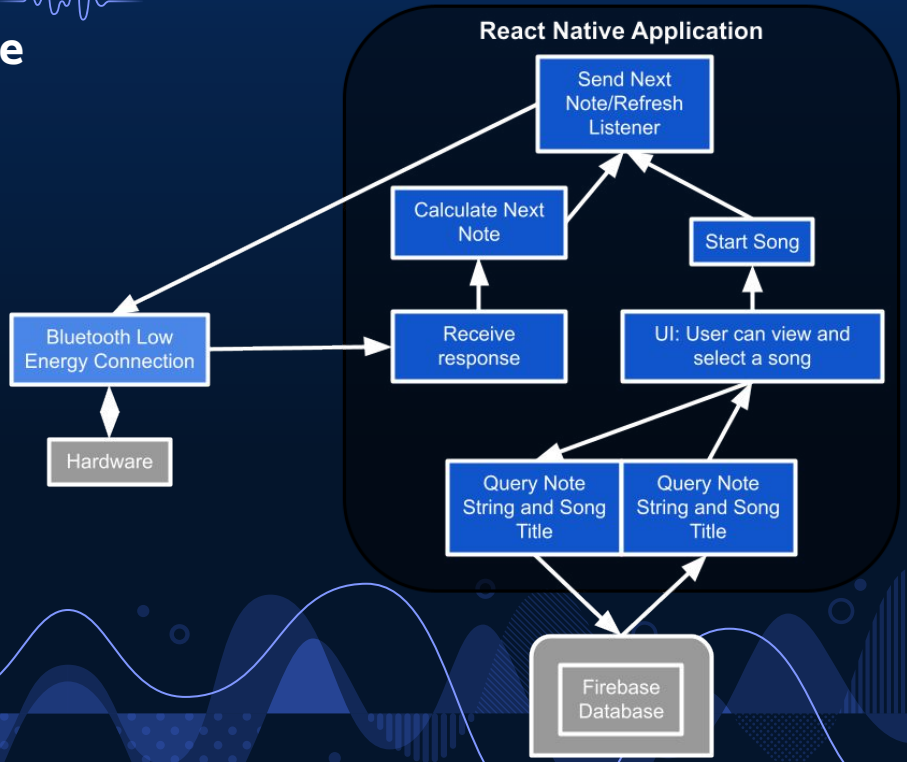
Implementation Architecture: Software - Kaitlyn

Application is created through **React-Native**

Song Storage done through a **Firebase Database**

Hardware–Software connection done through **Bluetooth Low Energy**

User starts the song and starts looping through sending notes and receiving a response from Hardware



Key Contributions - Stuart

Primary Role: Responsible for note detection

Step 1. Read the physical kalimba vibration frequencies

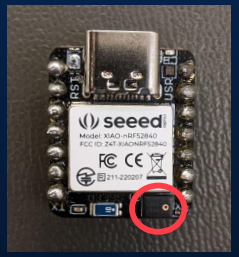
Explored 3 different pick up options

Step 2. Identify the frequencies present

Implemented a Fast Fourier Transform (FFT)

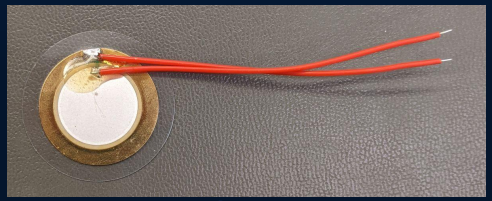
Step 3. Function returns array of top five notes detected

Pulse Density Modulated (PDM) Microphone



Condenser Microphone

Piezoelectric Pickup



Key Contributions - Stuart

Note Detection Decision Matrix

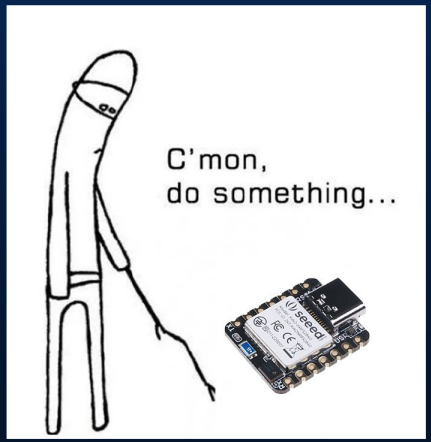
Note Detection Strategy	Note Detection Accuracy	Electrical Noise Rejection	Ambient (Room) Noise Rejection	Multinote Recognition	Implementation Ease	Total
Condensor Mic	8	7	8	7	5	35
PMD Mic	8	10	8	7	10	43
Piezoelectric Pickup	9	3	10	7	3	32

Secondary Role: Assisted with full system integration (e.g. battery, soldering, part orders)

Key Contributions - Isaac

Primary Role: Implementing Functionality to the Microcontroller

- Bluetooth communication to software
  - Change state processes depending on set characteristic values
- Setting the lights according to the notes that need to be played
- Referencing FFT function when searching for user input
- Communication point with the Software team to ensure a successful bridge when we joined forces second semester.
- Getting BLE to work involved both hardware and software teams to get together for troubleshooting. Held us up the longest, also a catalyst for pushing towards a battery powered device.



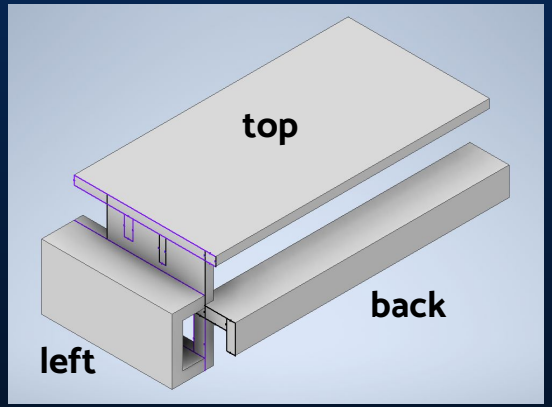
Secondary Role: LED Configuration Assembly/ Soldering Work

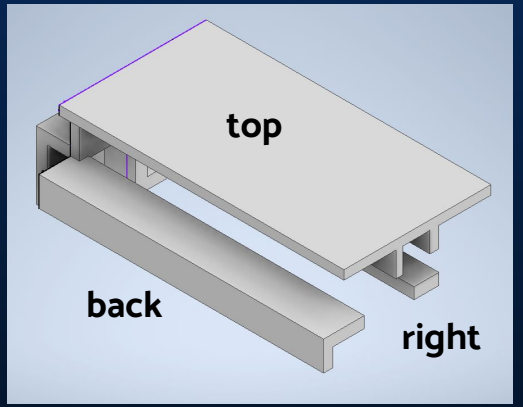
- Helped select the best LEDs to integrate into the design
- Solder work connecting lights in series (tedious work)

Key Contributions - Eileen

Device mount design and construction

- Autodesk Inventor
- Makerbot Printing





Problems and Solutions

- Must fit multiple kalimbas
  - Generalized design
- Device breaking or warping
  - Simplified and strengthened design
- Disrupted resonance of kalimba
  - Rubber Bands used for dampening

Key Contributions - Kaitlyn

Bluetooth Low Energy Connection through the Application

Accomplished

- Connect to the Hardware device
- Read/Write to the characteristics of the device
  - Updating the song mode
  - Reading the response (correct not or not)
  - Sending the next note to be played

Challenges/Solutions

- A large learning curve
- Some setbacks to the react-native BLE library
  - Auto connection, limited data types
  - Mismatched data types between HW/SW

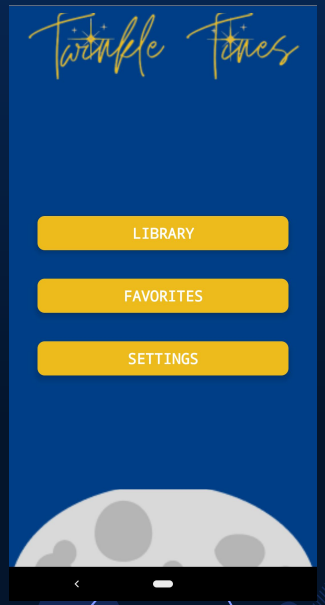
Key Contributions - Andrew

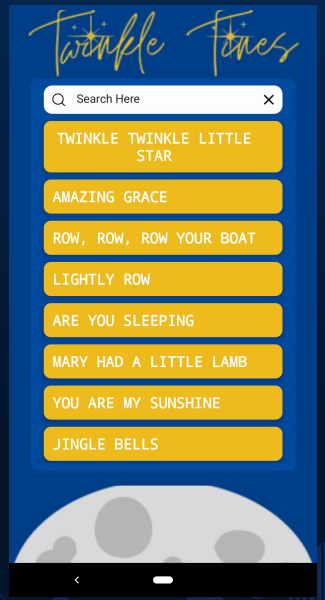
- Helped implement the **backend Firebase implementation** by using the database we created to store information about the songs.
- **Set up querying** from the frontend to the backend in the Library screen by which users can see all of the songs listed in the library.
- Helped **route the notes** from firebase database, and pull the notes when the user selects a specific song, and then routing through to our bluetooth files so the user will be able to get each individual note one at a time.
- **Created unit tests** using Jest and a mock database.

```
const fetchSongs = async () => {
  try {
    const songList: Song[] = [];
    const querySnapshot = await getDocs(collection(db, 'songs'));
    querySnapshot.forEach((doc) => {
      songList.push({ id: doc.id, ...doc.data() } as Song);
    });
    setSongs(songList);
  } catch (error: any) {
    console.error('Error fetching songs:', error.message, 'Error code:', error.code);
```

Key Contributions - Mesa

- **Work Accomplished**
  - UI implementation
- **Firebase Database**
  - Added songs to the firebase
  - Query the database for favorites
- **UI Implementation**
  - Design





Future Work

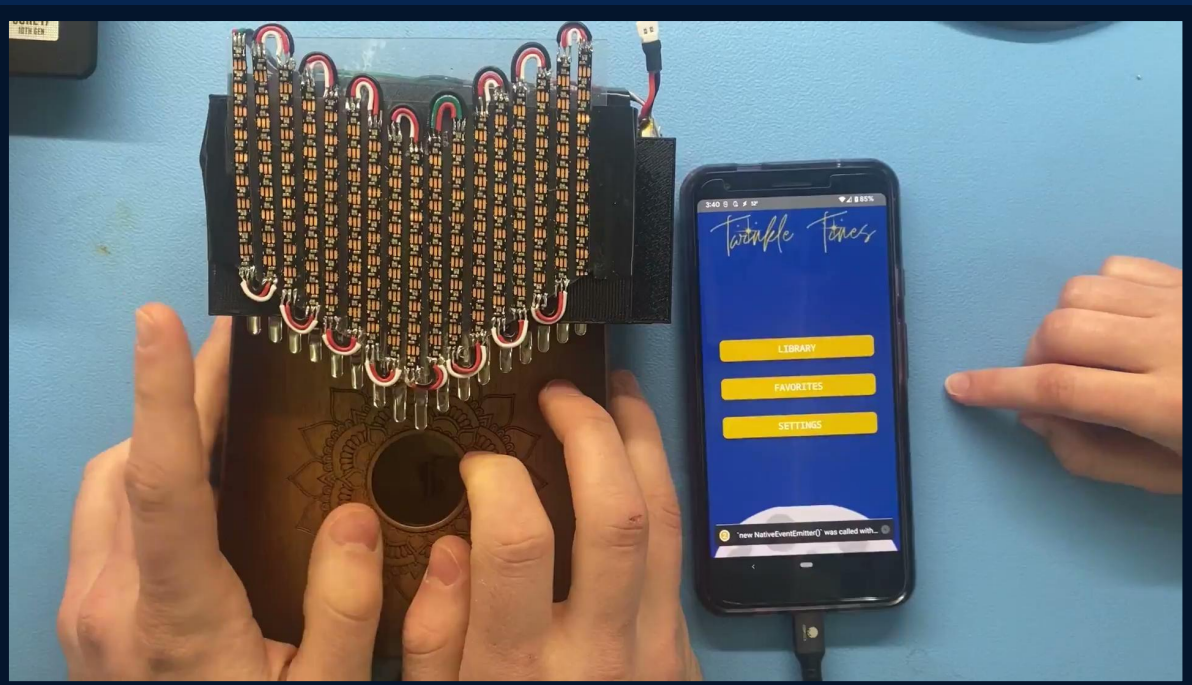
- Tweak Bluetooth auto-connection
  - Create a background worker for BLE to allow the connection between screens
- Add users to the database
  - Add users to the database so they can have their own unique favorites library
- Waterfall Mode
  - Code that would replace the current existing code and would show the incoming notes as lights falling down the LED canvas as it approaches the time it is to be played. Similar to Guitar Hero.
  - When note reaches the bottom-most LED, we check the FFT to see if the user played the note, then we would notify the app if the note was played successfully or not.
  - Incoming notes would be a continuous stream.
- More LED Color Variation
  - When microcontroller checks user's played note for the actual note, we then turn the bottom LED either green or red if note was played correctly or not.

Conclusion

- Goal
  - Create a device that makes it easy for anyone to play songs on a kalimba without having to know anything about music
- Accomplishments
  - We were able to create a prototype where a user can choose a song to play, on our application, that will allow the user to see what note needs to be played by lighting up the correct tine on our device
- Future
  - We would like to see this project go further as we see it as something that can always be improved on
  - When tasks are accomplished, new needs come about

Questions?


lemeiyi
KaliMba


Twinkle Tunes
LIBRARY
FAVORITES
SETTINGS