



University of Colorado  
Boulder

# **Lightning Bolts Present MusicVerse**

**CSCI 3308  
Summer 2023**



# Sean, Max, Mark, Yuzhou

## MusicVerse



University of Colorado **Boulder**

# Agenda

- Vision
- Tools
- Challenges
- Demonstration of your project
- Q&A



# MusicVerse

An online collection of easy to read sheet music for a variety of genres

## Vision

- To provide quality sheet music for all the world's musicians

## Motivation

- Developing proficiency in web-based music score rendering
- Creating a valuable platform for musicians
- Overcoming challenges and completing a complex project

# Tools

- [Github](#) - mark
- [Render](#) - yuzhou
- [Trello](#) - mark
- [Slack](#) - Sean
- [VS Code](#) - sean
- [pytest](#) - sean
- [Flask](#) - sean
- [Vite](#), [React.js](#), [Tailwind css](#) - yuzhou
- Pdoc, SwaggerUI - yuzhou
- [Zoom](#) - max

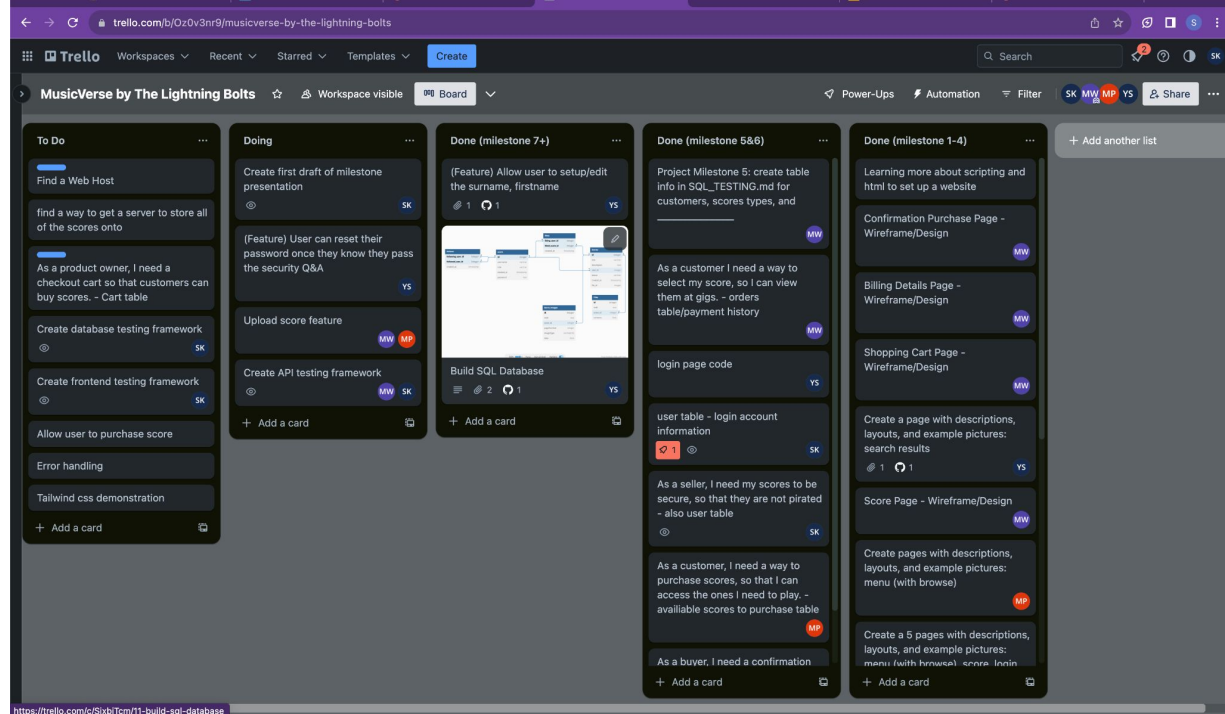




# Trello

## Backlog

- User Stories
- Iterations
- Feature Documents
- Task Assignment

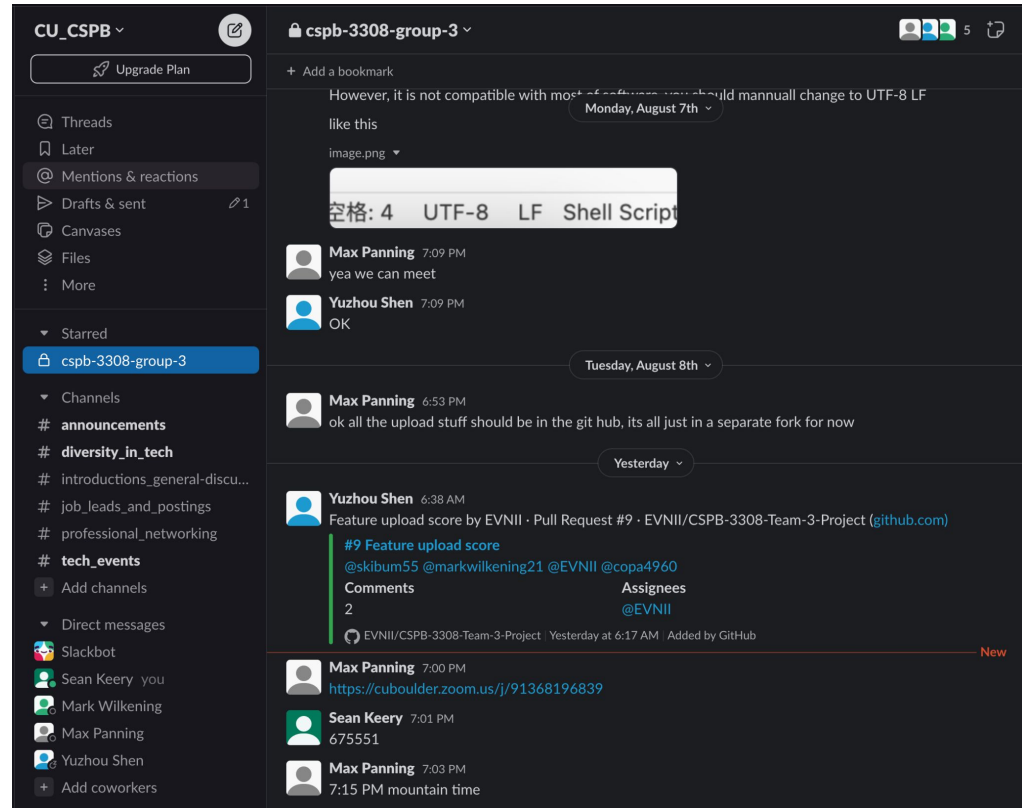


<https://trello.com/b/Oz0v3nr9/musicverse-by-the-lightning-bolts>

# Slack

## Chat

- Offline Comms
- File Sharing
- Historical Record

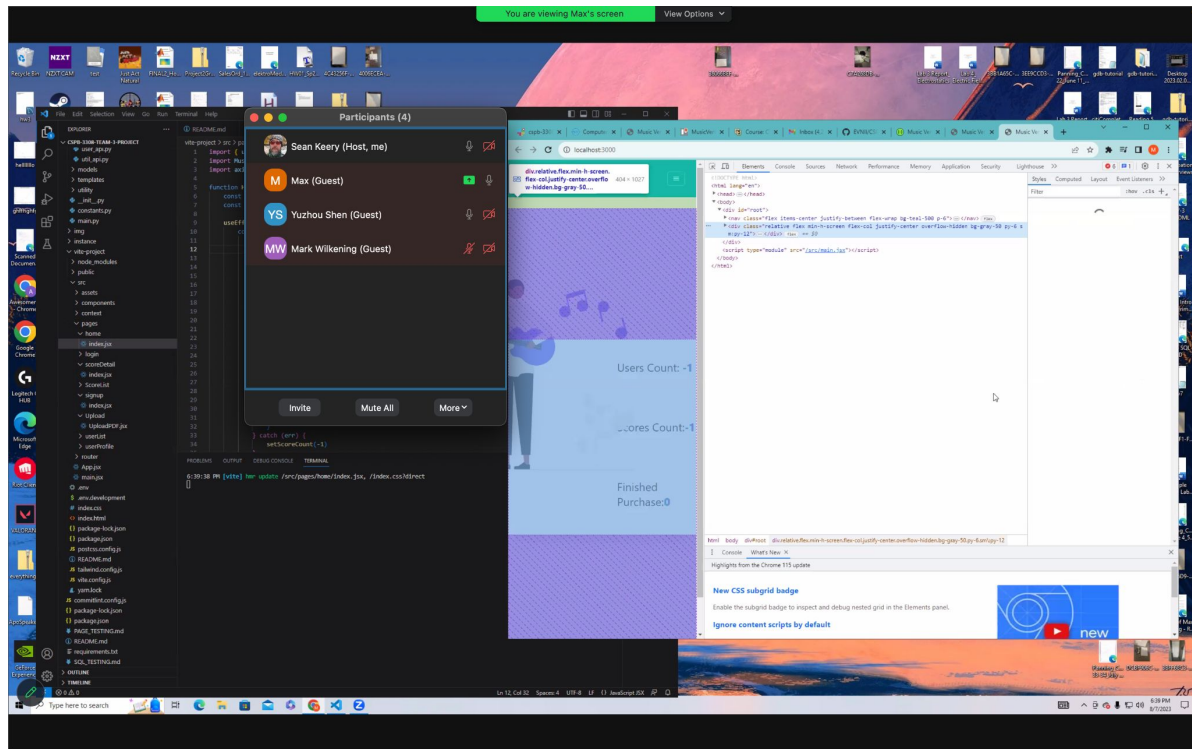


<https://cu-postbacc.slack.com>

# Zoom

## Real Time Collaboration

- Standup
- Code Review
- Demo Walk Through





# Github

## Version Control

- Track Changes
- Approve Updates
- Collaborative Coding
- Merging/Branches



The screenshot displays the GitHub web interface for a repository named 'CSPB-3308-Team-3-Project'. The left sidebar shows the file tree with folders like '.github', '.husky', 'api-testing', 'app', 'api', 'models', and 'templates'. The 'api' folder is expanded, showing files: '\_\_init\_\_.py', 'score\_api.py', 'user\_api.py', and 'util\_api.py'. The 'util\_api.py' file is selected and its content is displayed in the main area. The code is a Python file defining two REST API endpoints: '/user\_count' and '/score\_count'. Each endpoint has a corresponding class, 'UserCount' and 'ScoreCount', which inherit from 'Resource'. Both classes have a 'get' method that queries a database for active users or scores and returns the count. The interface includes tabs for 'Code', 'Blame', and 'Raw', and a search bar at the top.

```
9 from app.models import db, User, Score
10
11 util_ns = Namespace('util', description='Utility's Quesring')
12
13
14 @util_ns.route('/user_count')
15 class UserCount(Resource):
16     """'UserCount' support GET method to show the number of users"""
17
18     @util_ns.doc('get the number of users')
19     def get(self):
20         """Query User Counts"""
21         counts = db.session.query(User.active).filter_by(active=True).count()
22         return counts
23
24
25 @util_ns.route('/score_count')
26 class ScoreCount(Resource):
27     """'ScoreCount' support GET method to show the number of scores"""
28
29     @util_ns.doc('get the number of scores')
30     def get(self):
31         """Query Score Counts"""
32         counts = db.session.query(Score).count()
33         return counts
```

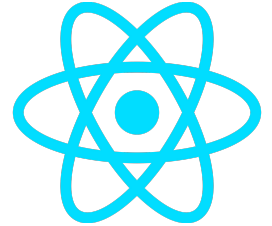


# FrontEnd Toolchain:

Vite: Build the environment for React.js, etc



React.js: Javascript Single Page App Framework



Tailwind css: A CSS Framework providing Atomic css classes



# VSCode

## Integrated Development Environment

- CSEL Jupyter
- Desktop
- Github Codespaces

```
File Edit Selection View Go Run Terminal Help
CSPB-3308-Team-3-Project - Visual Studio Code

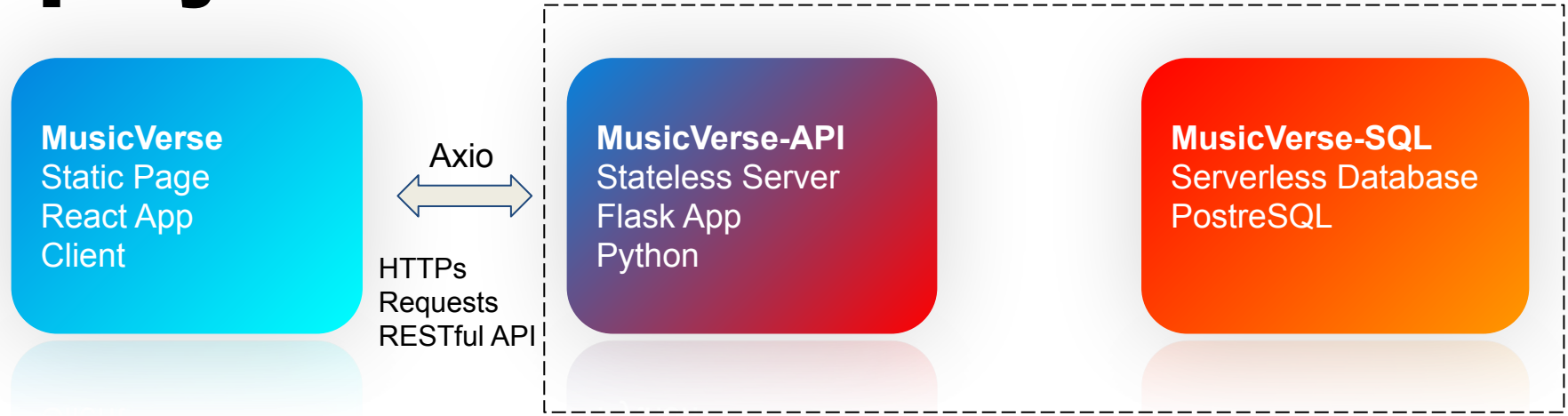
EXPLORER
CSPB-3308-TEAM-3-PROJECT
├── user_api.py
├── util_api.py
├── models
├── templates
├── utility
├── _init_.py
├── constants.py
├── main.py
├── img
├── instance
├── vite-project
├── node_modules
├── public
├── src
├── assets
├── components
├── context
├── pages
├── home
├── login
├── scoreDetail
├── index.jsx
├── ScoreList
├── signup
├── index.jsx
├── Upload
├── UploadPDF.jsx
├── userList
├── userProfile
├── router
├── App.jsx
├── main.jsx
├── .env.development
├── index.css
├── index.html
├── package-lock.json
├── package.json
├── postcss.config.js
├── README.md
├── tailwind.config.js
├── vite.config.js
├── yarn.lock
├── commitlint.config.js
├── package-lock.json
├── package.json
├── PAGE_TESTING.md

1 README.md > ## Project title: "MusicVerse" > ## FrontEnd Part > ## Run
working prototypes to have a finished project.
61
62 ### Project Tracking Software link (Trello is most common):
63 https://trello.com/w/thelightingboltspsb3308group3
64
65 ## BackEnd Part
66
67 Render Deployment: https://musicverse-api.onrender.com/SwaggerUI/
68 ## Setup python Environment.
69
70 ```shell
71 pip install -r requirements.txt
72 ```
73
74 ### Run
75 ```shell
76 export FLASK_APP=app
77 export FLASK_ENV=development
78 export MV_DB_CONNECT="<connection str>"
79 flask run
80 ```
81
82 ## Swagger UI
83 The Back end integrated with Swagger UI to check the api usage.
84 Can access http://127.0.0.1:5000/SwaggerUI/
85
86 ## FrontEnd Part
87
88 Render Deployment: https://musicverse.onrender.com/
89
90
91 ## Setup Environment
92 ```shell
93 cd vite-project
94 npm install
95 npm install
96 ```
97
98 ### Run
99 ```shell
100 echo "VITE_API_URI=http://127.0.0.1:5000" > .env.development
101 npm run dev
102 ```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
VITE v4.4.6 ready in 1125 ms
→ Local: http://localhost:3000/
→ Network: use --host to expose
→ press h to show help
PS C:\Users\mname\OneDrive\Desktop\CSPB-3308-Team-3-Project\vite-project> echo "VITE_API_URI=http://127.0.0.1:5000" > .env.development
```



# Deploy It On Cloud - Render

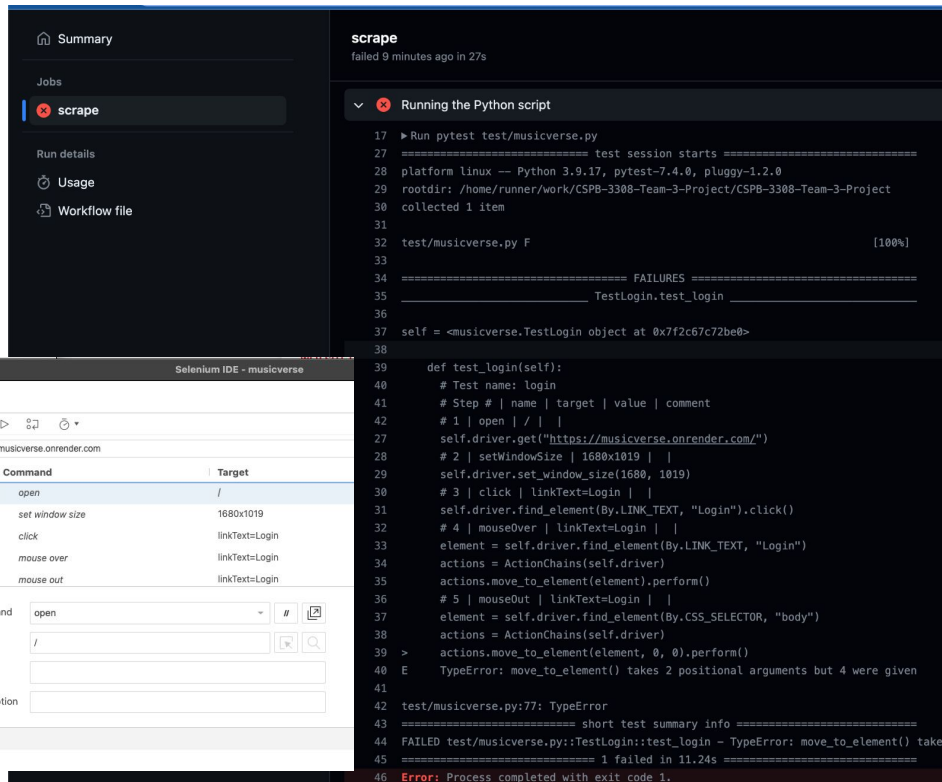
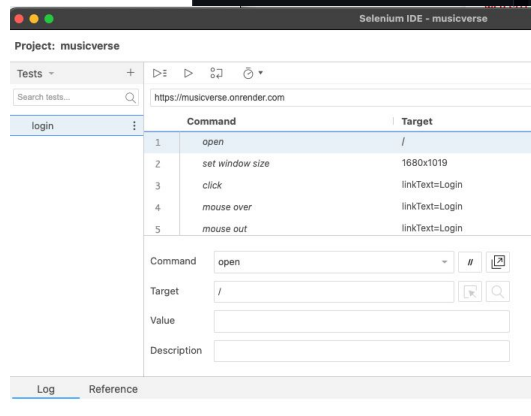


NAME	STATUS	TYPE	RUNTIME	REGION	LAST DEPLOYED ↓
🌐 MusicVerse-API	● Deploy succeeded	Web Service	Python 3	Oregon	a day ago ...
📁 MusicVerse	● Deploy succeeded	Static Site	Static	Global	a day ago ...
🗄 MusicVerse-SQL	● Available	PostgreSQL	PostgreSQL 15	Oregon	4 days ago ...

# pyTest

## Unit, User and Database Testing

- [Selenium Recorder](#)
- [Selenium](#)
- [Github Workflows](#)




# Auto Documentation:

## Pdoc: For Python

[Module List – pdoc 14.0.0 \(evnii.github.io\)](#)

Available Modules

- app.api
- app.models
- app.models.db
- app.utility



Generate Via Github Action

## SwaggerUI: For RESTful APIs


[Music Verse API](#)  
[\(musicverse-api.onrender.com\)](#)


Launch With API server













**Music Verse API** <sup>1.0</sup>

[ Base URL: / ]  
[/swagger.json](#)

Music Verse RESTful API

[Authorize](#) 

**user** User related operations 

GET	/api/user/account/	Get all user information	 
PUT	/api/user/account/{user_id}	Update user information with ID	 
GET	/api/user/account/{user_id}	Get user information with ID	 
POST	/api/user/login	Login user	 
GET	/api/user/login_info	Example route that requires login	 
POST	/api/user/signup	Create new user	 



# Challenges

- Test Automation
- Task Assignment/To-Do
- Version Control
- Encoding Issue



# Demo





# Q&A

