music mirror

Team B3: Thomas Lee, Luke Marolda, and **Matt Hegi**

Use-Case

- A comprehensive speaker attachment that seamlessly manages queuing, song recommendations, and crowd engagement
- Users steer the system through a distributed web app that hosts a suite of song request and consensus voting capabilities

Existing Solutions

- Current systems are singular - they focus on one person having full control. We democratize the event listening experience for uniform enjoyment

Areas

- Software Systems, Machine Learning, Hardware Systems

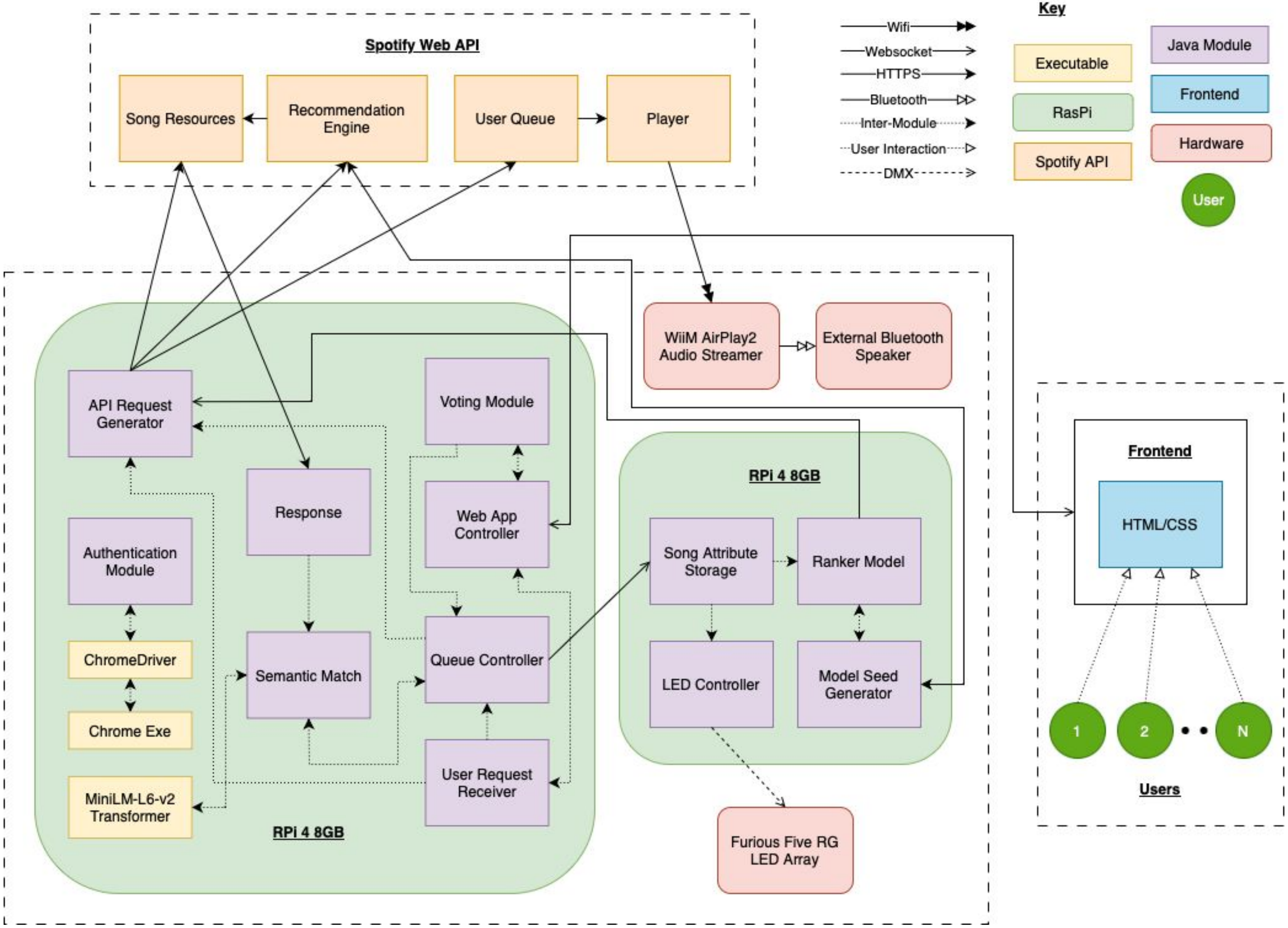
Design Requirements

Requirement	**Metric**	**Status**
Mount to any speaker	**Bluetooth** and **AUX**	✅
Song request formats	**3** distinct request formats	✅
User requests to queue	Reflected within **1 second**	✅
Manage concurrent users	**100-150** users	✅
Support live user feedback	**Vetoes** and **Likes**	✅
Easily usable mobile website	Onboarded in **<1 minute**	✅
Light strobing effects	**Transitions** with song	🟧

music mirror

Design Requirements (Improvements)

Requirement	**Metric**	**Status**
Endless queue	Queue is **never** empty	✅
Enhanced recommendation algorithm	**Finer-tuned** than Spotify	✅
Weighted session recommendations	Weighted by **user likes**	✅
Support volume adjustment	**Button** interface	✅



music mirror

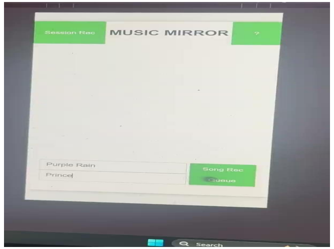
Ethical Considerations

Category	**Problem**	**Solution**	**Status**
Health	Unsafe operating volumes	Easy volume adjustment with button press	✅
Health	Unsafe light strobing	Max intensity: 225 (88%) Max frequency: 65ms	✅
Safety	User data security	Secure storage + data invisible to other users	✅
Welfare	Vulgar music content	Prohibited use of certain words when queueing	🟧

Public Demonstration Solution

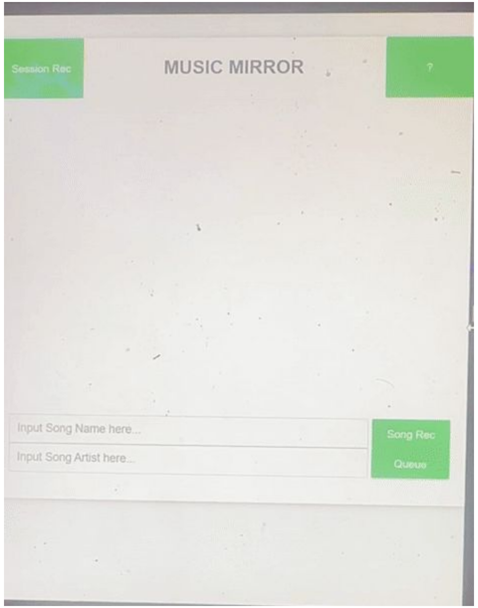
1. System boot-up 🔁
2. Simple queueing functionality by song name + artist
3. Queue scheduler mechanism + endless queue
4. Light strobing and effects ⚡
5. Song similarity recommendations
6. Likes and dislikes ➡️ veto functionality
7. User keep-alive functionality
8. Weighted session recommendations
9. Open up to the public!

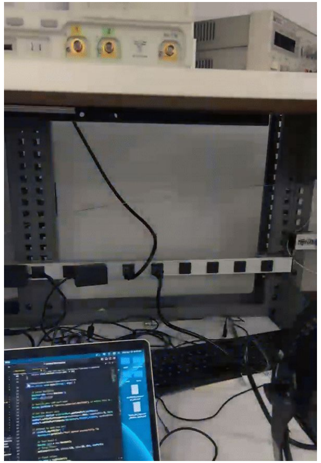
🚨 Full functionality with concurrent users 🚨



Purple Rain

Prince





Testing, Verification, and Validation

Latency	Web App to System: measure latency for a single time-stamped Play Song request to be reflected on queue (< 1 sec)
Capacity	Queue: verify that all Main RPi queue can maintain 100+ songs without running out of memory, and perform operations under max latency User Network: verify that Main RPi can accept ambiguously timed requests from 100-150 concurrently online users and maintain ordering

System Latency ✅

- Measured (with timestamps) direct queue request latency = 102 msec (20 trials)
- Recommendation request latency = 6.349 sec (20 trials)

Queue ✅

- Manually constructed queue session with 100 songs (direct + recommendations) and maintained performance without running out of memory (1 trial)

User Network 🟧

- Developed a test to test 150 concurrent users. Have yet to test.

Testing, Verification, and Validation

Accuracy	Queue: use script / live tests to issue song requests in a certain order, verify that they appear in that same order on system (and then back on web app) Resources: 80% accuracy in match between user input and Spotify resource Lighting: use hard coded light script to verify that we can control each light channel independently and to do the intended color & strobing

Queue ✅

- Conducted **2** live tests with 20-30 concurrent users scattered across campus, to test robustness of concurrency handling and queue ordering

Semantic Match ✅

- Expected Matches: Avg Similarity = **83.6%**, Max Similarity = **100%**, Min Similarity = **67.1%**
- Expected Failures: Avg Similarity = **46.2%**, Max Similarity = **57.3%**, Min Similarity = **18.0%**

Lighting 🟧

- Stress tests for transmitting DMX data frames: system capable of **<100ms** response time intervals to new control signals for **all 10 channels** on SlimPAR PRO Q USB

Testing, Verification, and Validation

music mirror

User Experience	Web App: measure average time to onboard new users, poll on 1-5 scale for ease of use and input responsiveness Recommendations: generate recommendations based on our compound model, poll users on 1-5 scale for quality of recommendations and compare to their ratings for generic Spotify recommendations

Web App ✅

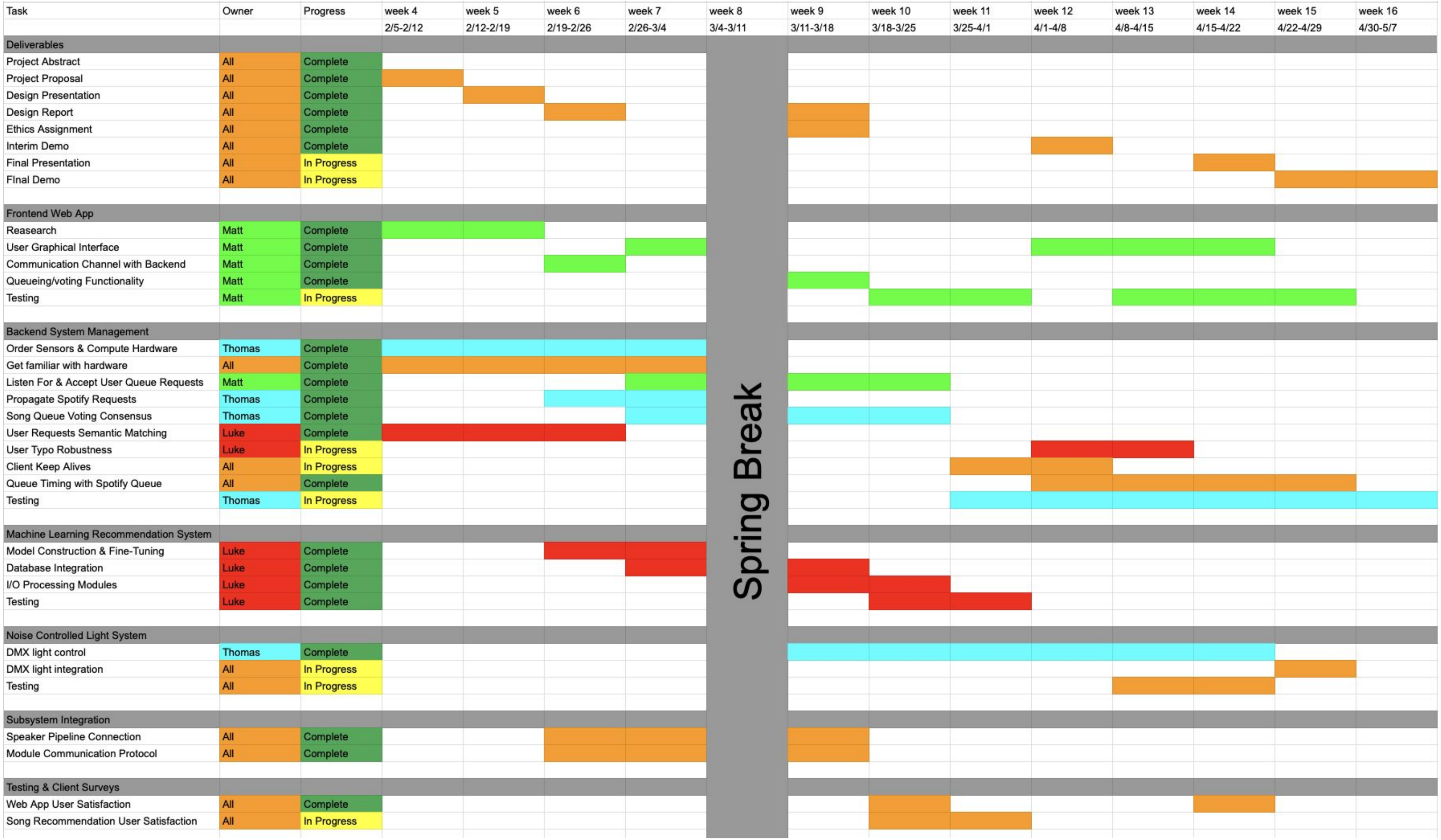
- Onboarding time: interviewed 10 participants: Average onboard time = **48.4 seconds**
    - Minimum onboard time = **21 seconds**, Maximum onboard time = **83 seconds**
- Average ease of use rating = **4.5/5**

Recommendations 🟧

- In the process of surveying users on the quality of song recommendations
    - Primarily concerned with **single song recommendations**, and comparing our enhanced recommendations with the bare Spotify endpoint recommendations

Engineering Tradeoffs

Recommendation Generation	Chose a simpler/more efficient models that fit our use case
Semantic Match	Prioritized whole word accuracy over user typos
Complexity vs Usability	Enhance features while maintaining a simple, intuitive user interface
Lighting System	Light intensity & strobing frequency limit for user health



Task	Owner	Progress	week 4	week 5	week 6	week 7	week 8	week 9	week 10	week 11	week 12	week 13	week 14	week 15	week 16
			2/5-2/12	2/12-2/19	2/19-2/26	2/26-3/4	3/4-3/11	3/11-3/18	3/18-3/25	3/25-4/1	4/1-4/8	4/8-4/15	4/15-4/22	4/22-4/29	4/30-5/7
Deliverables							Spring Break								
Project Abstract	All	Complete													
Project Proposal	All	Complete	🟧												
Design Presentation	All	Complete		🟧											
Design Report	All	Complete			🟧			🟧							
Ethics Assignment	All	Complete						🟧							
Interim Demo	All	Complete									🟧				
Final Presentation	All	In Progress											🟧		
FInal Demo	All	In Progress												🟧	🟧
Frontend Web App															
Reasearch	Matt	Complete	🟩	🟩											
User Graphical Interface	Matt	Complete				🟩					🟩	🟩	🟩		
Communication Channel with Backend	Matt	Complete			🟩										
Queueing/voting Functionality	Matt	Complete						🟩							
Testing	Matt	In Progress							🟩	🟩		🟩	🟩	🟩	
Backend System Management															
Order Sensors & Compute Hardware	Thomas	Complete	🟦	🟦	🟦	🟦									
Get familiar with hardware	All	Complete	🟧	🟧	🟧	🟧									
Listen For & Accept User Queue Requests	Matt	Complete				🟩		🟩	🟩						
Propagate Spotify Requests	Thomas	Complete			🟦	🟦									
Song Queue Voting Consensus	Thomas	Complete				🟦		🟦	🟦						
User Requests Semantic Matching	Luke	Complete	🟥	🟥	🟥										
User Typo Robustness	Luke	In Progress									🟥	🟥			
Client Keep Alives	All	In Progress								🟧	🟧				
Queue Timing with Spotify Queue	All	Complete									🟧	🟧	🟧	🟧	
Testing	Thomas	In Progress								🟦	🟦	🟦	🟦	🟦	🟦
Machine Learning Recommendation System															
Model Construction & Fine-Tuning	Luke	Complete			🟥	🟥									
Database Integration	Luke	Complete				🟥		🟥							
I/O Processing Modules	Luke	Complete						🟥	🟥						
Testing	Luke	Complete							🟥	🟥					
Noise Controlled Light System															
DMX light control	Thomas	Complete						🟦	🟦	🟦	🟦	🟦	🟦		
DMX light integration	All	In Progress												🟧	
Testing	All	In Progress										🟧	🟧		
Subsystem Integration															
Speaker Pipeline Connection	All	Complete			🟧	🟧		🟧							
Module Communication Protocol	All	Complete			🟧	🟧		🟧							
Testing & Client Surveys															
Web App User Satisfaction	All	Complete							🟧				🟧		
Song Recommendation User Satisfaction	All	In Progress							🟧	🟧					