CS 3354 Software Engineering
Final Project Deliverable 2

Noteworthy

Ana Alvarez
Sneha Bista
Emma Hockett
Katrina Lee
Nuvia Mata
Lauren Anne Nicolas
Sarah Yakum

Table of Contents

1.	Delegation of Tasks	3
2.	Deliverable 1 Content	4
2.1.	Final Project Proposal	4
2.2.	Github Link	5
2.3.	Delegation of Tasks	5
2.4.	Software Process Model	6
2.5.	Software Requirements	6
2.6.	Use Case Diagrams	10
2.7.	Sequence Diagrams	13
2.8.	Class Diagrams	17
2.9.	Architectural Design	17
3.	Project Scheduling, Cost, Effort and Pricing, Project Duration and staffing	18
4.	Test Plan	24
5.	Comparison with similar designs	26
6.	Conclusion	27
7.	References	28
8.	Presentation Slides	29

1. **Delegation of Tasks**

The whole team will meet up and work on the project together, but the tasks will have a specific person in charge. Everyone can help and give feedback, but the person in charge has to make sure the specific task is completed in the end.

- Turning in everything - Katrina

Deliverable 1:

- Setting up GitHub repository
  - 1.1 - Create a GitHub account - Everybody
  - 1.2 - Create a repository - Sarah
  - 1.3 - Add all team members and the TA - Lauren
  - 1.4 - First commit/README - Katrina
  - 1.5 - project_scope commit - Nuvia
  - 1.6 - add url of repository into deliverable 1 report - Lauren
- Addressing feedback - Emma
- Software process employed in project - Ana
- Software Requirements
  - Functional Requirements - Katrina
  - Non-functional Requirements - Ana
- Software Process Model Description/ Explanation - Emma
- Use Case diagram (multiple) - Sneha
- Sequence Diagram (for each use case diagram) - Nuvia
- Class Diagram - Sneha
- Architectural Design - Sarah

Deliverable 2:

- Project scheduling - Nuvia
- Cost, effort, price estimation method - Ana
- Cost estimation of hardware, software, and personnel - Sarah
- Software test plan - Katrina
- Comparison of work with similar designs - Sneha
- Conclusion - Emma
- Presentation slides - Lauren

2. Project Deliverable 1 Content

2.1 <u>Final Project Draft Description</u>

We'll be creating an app/website that combines Rotten Tomatoes, Letterboxd, and music listening platforms all into one place. Our platform includes features like:

- **Home page** where at the top it shows what the user has recently listened to and as they scroll down, news about what's popular, newly released music, etc. Additionally, there will be buttons that take users to different pages like discover new music or visit their library.
- **Sign in page** where users can create an account to keep a record of the music they listen to and review. Signing up allows users to connect to an already existing music streaming platform, so users can play music, or if the user is just interested in making an account with us for discussion purposes.
- **Community page** where users can see critics' and audience's reviews, Billboard statistics, and what other users are listening to. This page also includes a general forum where users can make posts and react and reply to other posts as well (like Reddit). This same feature will also be under each artist's page so that way it's easier to find.
- **User profile page** where users can see their personal statistics for their music (top artists, top album, top songs, etc). They can also see their listening history, follow friends, and find all the reviews they have written. Additionally, this page will include:
    - Library of songs and albums where they can be filtered by favorite songs, albums, artists
    - Recommended albums, artists, and curated playlists
- **Music page** includes various features:
    - A search bar where users can search by genre, song, or artist.
    - Lists of songs that are sorted by recently released, trending, genre, and artists.
    - Each song has features like:
        - Users are able to make comments and highlight lyrics as well as see the most liked comment, lyric, and most highlighted lyric.
        - Song games like karaoke (only instrumental, no lyrics), guess that song (all the different variations: guess with emojis, guess with lyrics, guess with 1 second of the song), finish the lyric, etc.

Our team wanted to have a platform to share and read other people's opinions on music for songs that they are listening to. This lets people discover new music that might be similar to their own or gain additional insight from other listeners. We like the features of Receiptify, a website where users are able to see their top artists, genres, and songs. We also enjoy features of book review apps, such as Goodreads, that allow users to rate, leave reviews, and gain suggestions. We want to integrate several features within an actual music platform to make it more accessible for

people to see within one place. With this platform, we hope to make it a hub for music for users to be able to see music videos, post their own reviews, and assess their listening history.

Feedback: "A practical idea of a tool that promises a lot of potential use. In the final report, please make sure to include comparison with similar applications -if any-, make sure that you differentiate your design from those, and explicitly specify how. Fair delegation of tasks. You are good to go."

Addressing the feedback: In the Final Project Deliverable 2, one of the tasks is to compare the project with similar designs. In that task we will do as the feedback suggests we will compare the design with others and explicitly specify how Noteworthy differs from those.

2.2 GitHub Link

https://github.com/utd-stu/3354-NoteWorthy

2.3 Delegation of Tasks

The whole team will meet up and work on the project together, but the tasks will have a specific person in charge. Everyone can help and give feedback, but the person in charge has to make sure the specific task is completed in the end.

- Turning in everything - Katrina

Deliverable 1:

- Setting up GitHub repository
  - 1.1 - Create a GitHub account - Everybody
  - 1.2 - Create a repository - Sarah
  - 1.3 - Add all team members and the TA - Lauren
  - 1.4 - First commit/README - Katrina
  - 1.5 - project_scope commit - Nuvia
  - 1.6 - add url of repository into deliverable 1 report - Lauren
- Addressing feedback - Emma
- Software process employed in project - Ana
- Software Requirements
  - Functional Requirements - Katrina
  - Non-functional Requirements - Ana
- Software Process Model Description/ Explanation - Emma
- Use Case diagram (multiple) - Sneha
- Sequence Diagram (for each use case diagram) - Nuvia
- Class Diagram - Sneha
- Architectural Design - Sarah

Deliverable 2:

- Project scheduling - Nuvia
- Cost, effort, price estimation method - Ana
- Cost estimation of hardware, software, and personnel - Sarah
- Software test plan - Katrina
- Comparison of work with similar designs - Sneha
- Conclusion - Emma
- Presentation slides - Lauren

2.4 Software Process Model

As a group, we will be employing the agile development process model for Noteworthy. By using agile we are able to separate the work among the team members and then later combine them in an easier way than some of the other software models. By doing this we are essentially creating an increment as everyone completes their share of the work, and earning more and more complete versions of the final project. Using agile also allows for easier adaptation to change, which might occur frequently as we are all working toward the final deliverable. These changes might come from internal disagreements within the team, misunderstandings, external restrictions, or any other various reasons that come from both within the team or external to the team. However, the agile process is more flexible than some other models and allows for discussions, negotiations, and possible changes to be made effectively and quickly without risking the progress of the project at that time. Another advantage of using agile for this project is the emphasis on customer involvement throughout the process. As our project will be centered around user interaction, the customer involvement will provide feedback as the process evolves which allows for easier changes and clears any misunderstandings between customers and creators early in the process.

Overall, agile development is the best process model for the Noteworthy software and Noteworthy team. The main advantages include adaptation to changes, separation of work within the team, incremental deliveries, reduced overhead on planning, and customer involvement. Although these might not be ideal for some software, they will benefit Noteworthy and are the most effective use of resources for the project.

2.5 Software Requirements

Functional Requirements

1. The system must display popular and trending music suggestions on the user's home page.
2. The system must allow users to log into their account by entering their username and password.

3. Signed-in users must be able to connect their music streaming service accounts to play music.
4. Users must be able to view and post comments and reviews to a community forum.
5. Signed-in users must be able to view their personalized statistics, connections with other users, library of saved music, and a list of recommendations on their user profile page.
6. Users must be able to search and sort through the database of songs and albums by genre, title, and/or artist.
7. Users must be able to interact with songs by playing guessing games and commenting and highlighting specific lyrics.

Non-functional Requirements

- **Product Requirements**
- Usability Requirements:
    - The Noteworthy system shall provide a new user navigation training that takes at most 1 minute.
    - The Noteworthy system shall provide in-app help and documentation including tooltips, tutorials, and FAQs.
    - The Noteworthy system shall provide a screen reader for the application to be more accessible for visually impaired and blind users.
    - The Noteworthy system shall meet Web Content Accessibility Guidelines (WCAG) standards to ensure it is accessible to users with disabilities.
    - The Noteworthy system shall provide different language options for users to choose which language the application is in.
    - The Noteworthy system shall provide keyboard navigation for all functionality so users can interact with the system without a mouse or touch input.
- *Efficiency Requirements*
- Performance Requirements:
    - The Noteworthy system shall be able to load the user's requested data within 2 seconds for 95% of requests like searching for songs, creating playlists, and posting/editing forum posts.
    - The Noteworthy system shall have a 60Hz refresh rate
    - The Noteworthy system shall have a response time of 0.1 seconds or less.
    - The Noteworthy system shall load the user's library and playlist within 2 seconds.
    - The Noteworthy system shall support a minimum of 10,000 concurrent users streaming music without a noticeable decrease in audio quality.
    - The Noteworthy system shall support various audio file formats like MP3 and WAV

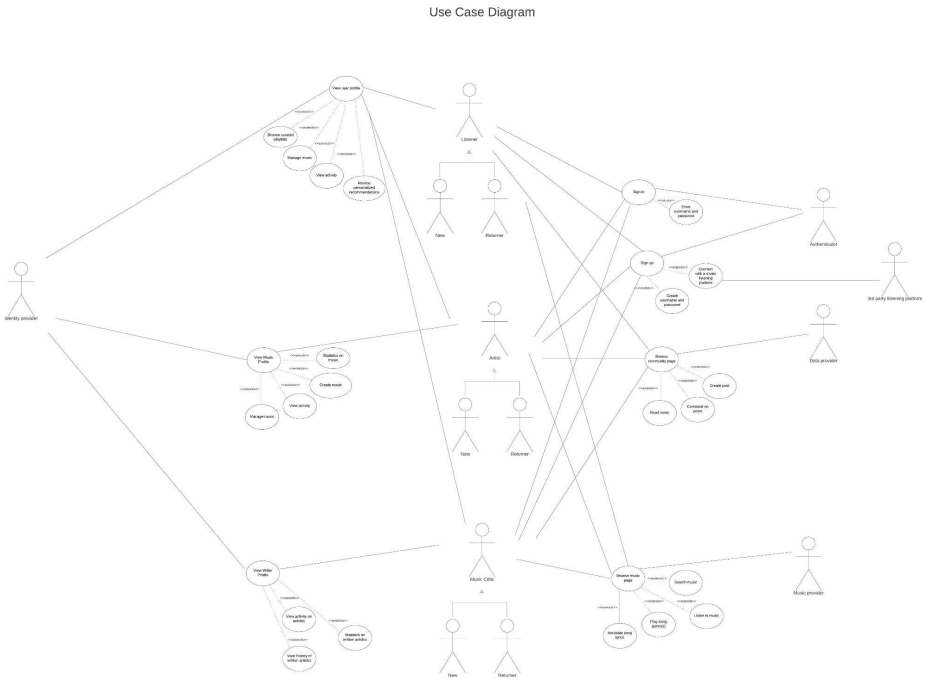
- Space Requirements:
    - Users shall have an image and attachment upload limit of 10MB per post.
    - The Noteworthy system shall support a maximum of 5,000 characters per forum post.
    - The Noteworthy application shall take up 500MB of memory on desktop and mobile.
    - The Noteworthy system shall contain forums on a cloud to limit user's space requirements.
    - The Noteworthy system shall have a music library so that users can access music.
    - The Noteworthy system shall optimize server resources to ensure that there is no excessive consumption of CPU, memory, or bandwidth.
- Dependability Requirements:
    - The Noteworthy system shall have a 99.99% uptime during any given month.
    - The Noteworthy system shall handle a 30% increase in user activity during peak times.
    - Data shall be backed up every 24 hours.
    - The web server shall run for about 8-9 months before failure and maintenance is required.
    - The mobile app shall run for about 3 months before failure maintenance is required.
- Security Requirements:
    - The Noteworthy system shall require a username and password each time the user is logging in.
    - The Noteworthy system shall have all users verify login credentials (email/phone number) when creating an account.
    - User data, including login credentials and personal information, shall be stored securely using encryption.
    - The Noteworthy system shall provide users with privacy settings and data sharing preferences.
- **Organizational Requirements**
- Environmental Requirements:
    - The Noteworthy system shall work seamlessly over high-speed and low-bandwidth networks connections.
    - The Noteworthy system shall work on the operating systems like Windows, Mac, and Linux
    - The Noteworthy system shall be compatible and usable on various devices like desktop, mobiles, and tablets.

    - The Noteworthy system shall support the latest versions of popular web browsers like Chrome, Firefox, Safari, and Edge.
    - The Noteworthy system shall work on mobile apps with an iOS and Android software
- Operational Requirements:
    - The Noteworthy system shall track user engagement and monitor platform usage for future improvements.
    - The Noteworthy system shall authenticate existing music streaming platforms accounts to transfer data and information.
    - The Noteworthy system shall authenticate profile types (like artist, user, or music critic) and give access levels depending on their type.
    - The Noteworthy system shall have a plan to address unexpected outages, security breaches, and other critical issues.
    - The Noteworthy system shall have a disaster recovery plan to address catastrophic events such as server failures or natural disasters.
- Development Requirements:
    - The system shall be constructed in about 5-6 months with an additional 2 months of testing.
    - The system budget shall be of ~$85,000, including licensing fees.
    - The Noteworthy system shall play music from the user's connected music streaming account.
- **External Requirements**
- Regulatory Requirements:
    - The Noteworthy system shall adhere to industry standards for music streaming and distribution.
    - The Noteworthy system shall comply with data protection regulations and copyright laws
    - The Noteworthy system shall track and renew music licensing agreements to ensure copyright compliance
    - Privacy policies and data handling practices shall continuously be updated to ensure compliance with privacy regulations.
- Ethical Requirements:
    - The Noteworthy system shall filter and moderate content to prevent spam, inappropriate/abusive content, and hate speech.
    - The Noteworthy system shall have a tagging system that allows artists to be credited on songs
    - The Noteworthy system shall have verified profiles for artists profiles to differentiate from imitation profiles.
    - The Noteworthy system shall have a tagging system to label explicit music.

    - The Noteworthy system shall have content moderation to restrict and remove content that has hate speech, harassment, or defamation.
- *Legislative Requirements*
- Accounting Requirements:
    - The Noteworthy system shall track music played so that the information can be sent back to the original music streaming platforms
    - The Noteworthy system shall compensate the artists for plays, downloads, and ad revenue.
- Safety/ Security Requirements:
    - The Noteworthy system shall have a real-time monitor for system performance and security breaches.
    - Users that are creating an account and not connecting through their music streaming platform shall create strong unique passwords for added security.
    - The Noteworthy system shall implement data encryption between users' devices and the server to ensure confidentiality and integrity of data.
    - The Noteworthy system shall comply with relevant regulations and standards (like copyright laws and accessibility requirements).
    - The Noteworthy system shall implement CAPTCHA or other verifications mechanisms to prevent automated account creation and spam.

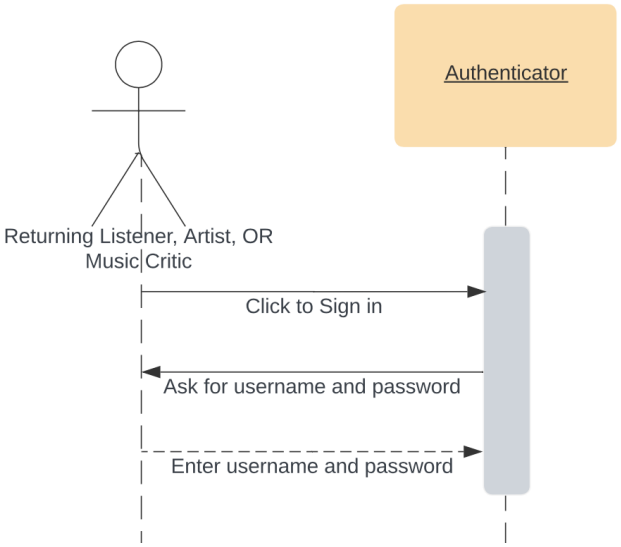
2.6 Use Case Diagram - for better viewing

Use case diagram

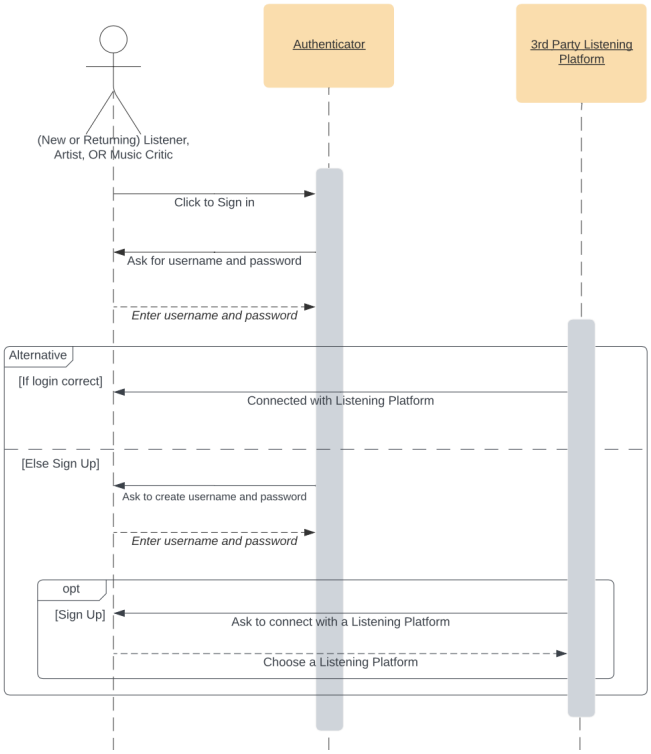


2.7 Sequence Diagram

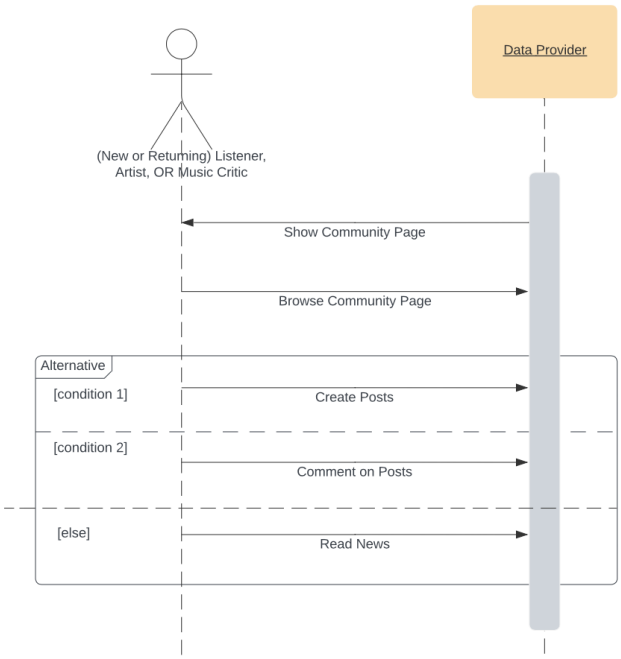
Sequence Diagram for Sign in:



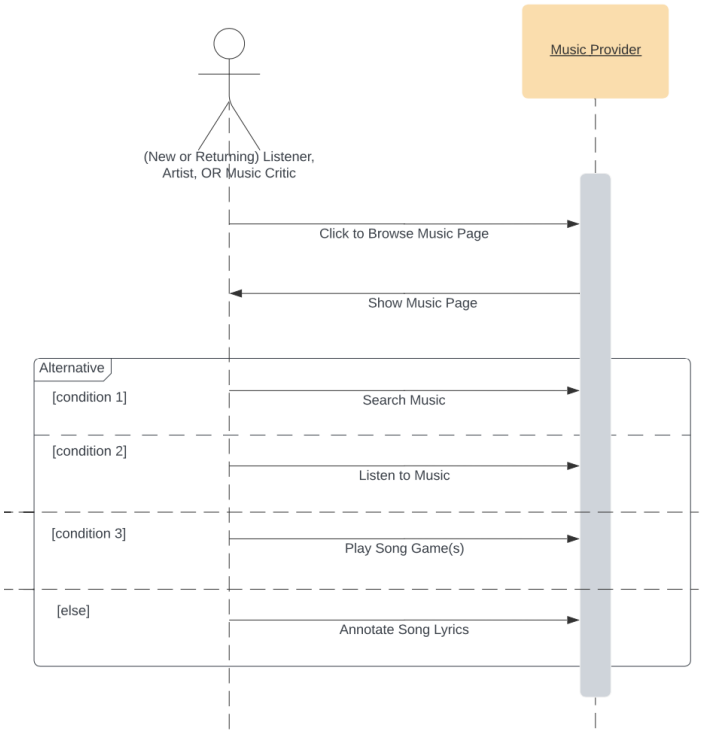
Sequence Diagram for Sign up:



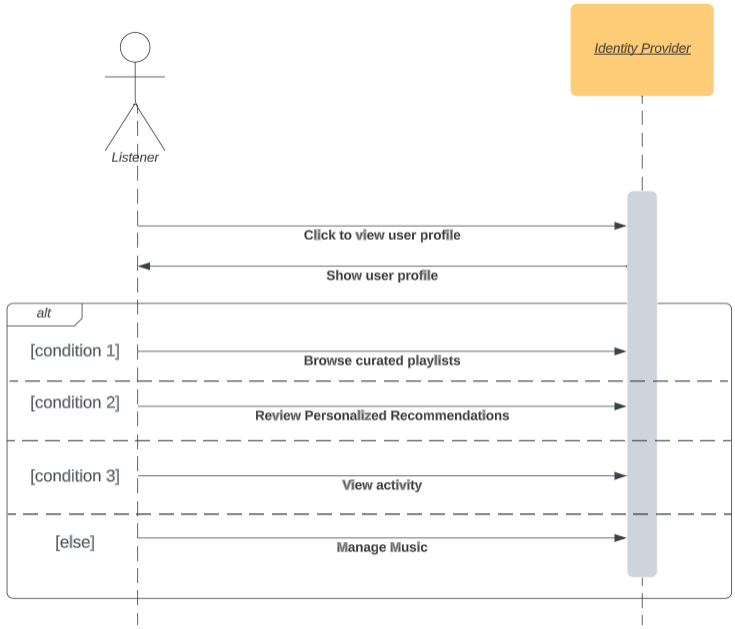
Sequence Diagram for Browse Community Page:



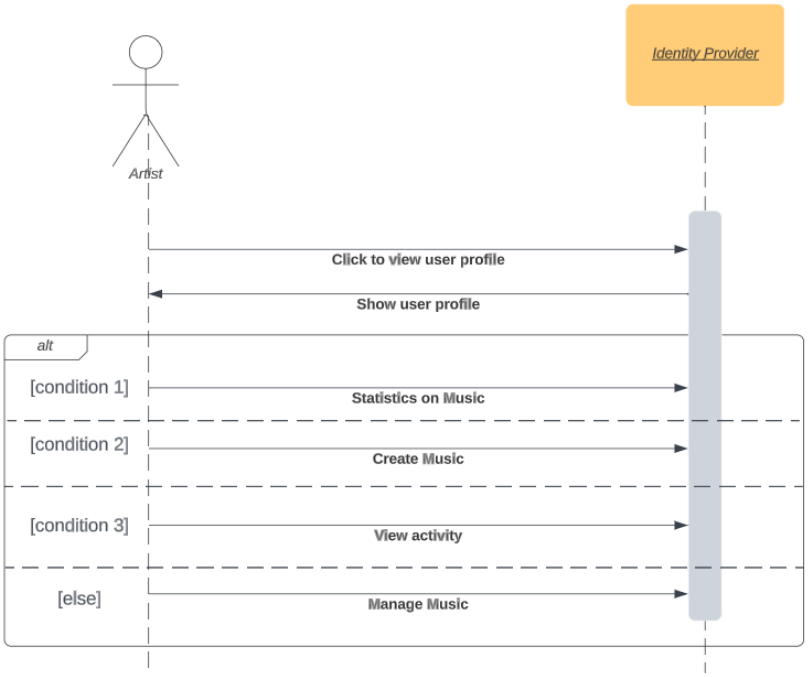
Sequence Diagram for Browse Music Page:



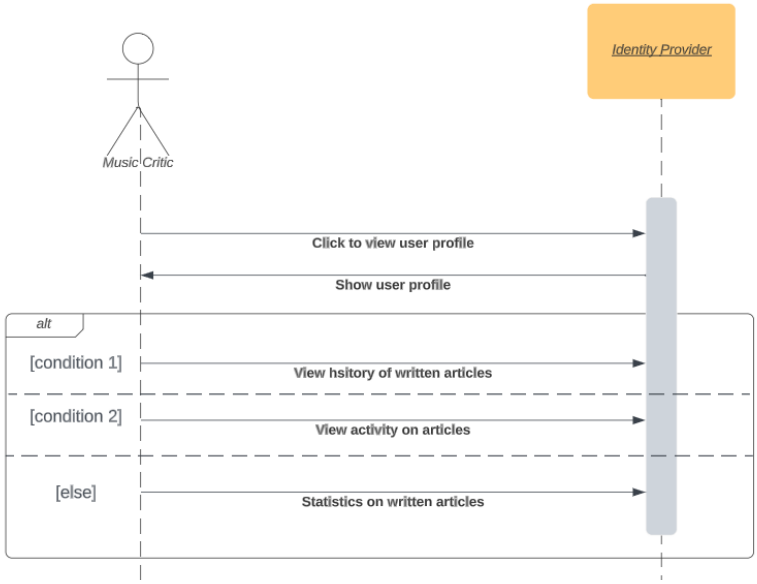
Sequence Diagram for View User Profile:



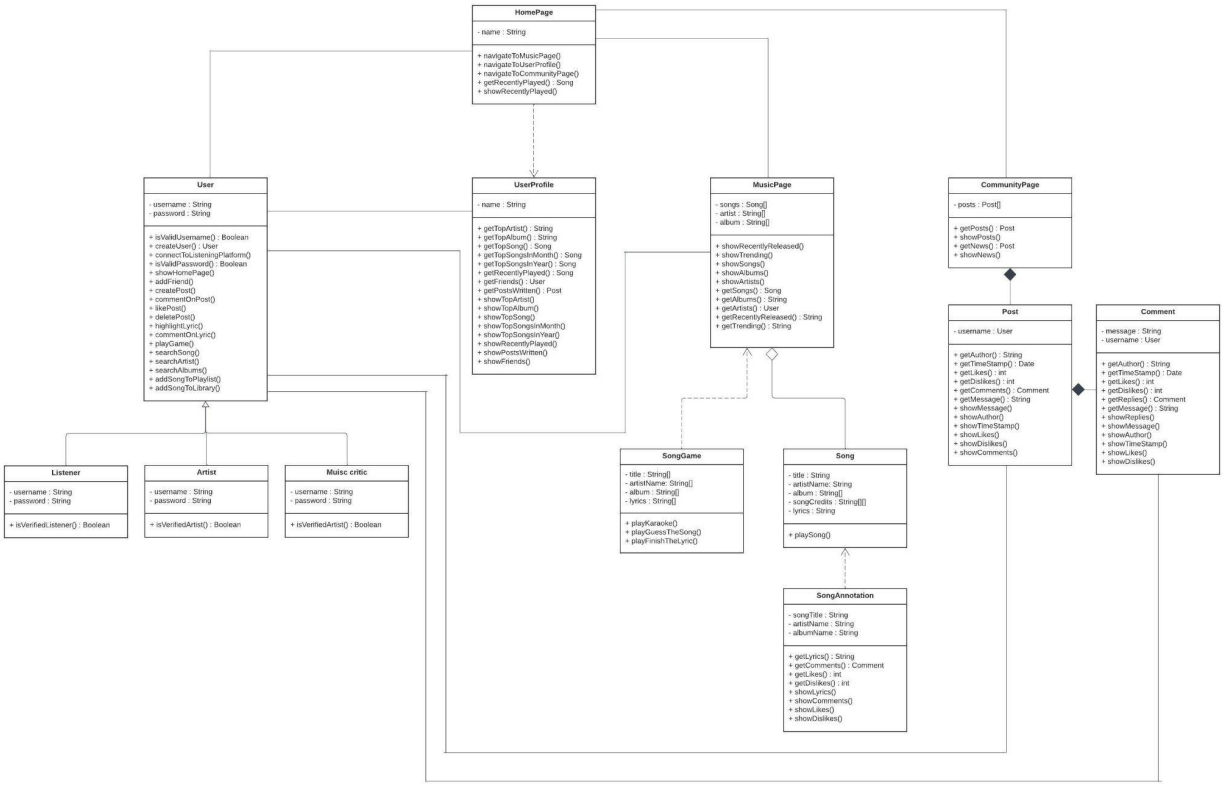
Sequence Diagram for View Music Profile:



Sequence Diagram for View Writer Profile:

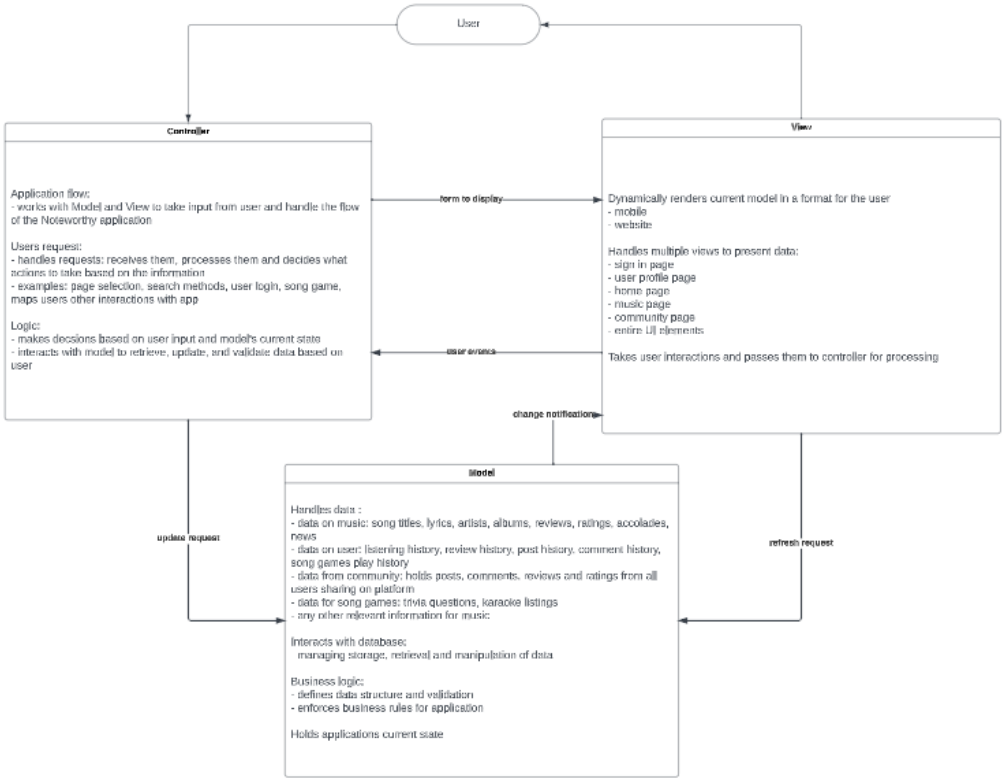


2.8 Class Diagram - for better viewing



2.9 Architectural Design

Using the Model-View-Controller pattern: link for better viewing



3. Project Scheduling, Cost, Effort and Pricing Estimation, Project Duration and staffing

After seeing other Software Development Project Schedules, it is clear that first you have to name your tasks and everything you have to do. Grouping them according to what they are will help the organization be more clear as well.

Groups of Tasks:

1. Planning
    a. Select Resources
    b. Research
2. Requirements Analysis
    a. Requirements review
    b. Requirements sign-off

    c. Security planning
    d. Gather all resources
3. Design
    a. Draft high level design document
    b. Proof of concept
    c. Design review
    d. Technical specifications
4. Development
    a. Test planning
    b. Integration planning
    c. Documentation
    d. Training plan
    e. Deployment planning
    f. Execution
5. Reporting
    a. Quality control
    b. Risk management
    c. Budget control
6. Implementation
7. Quality Assurance
8. Test
    a. Demo
    b. Issues resolution
    c. Improvements plan
9. Maintenance
10. Completion
    a. Documentation
    b. Finalize Everything
11. Launch

START DATE: Monday January 8, 2024 (holidays have passed; no distractions)
END DATE: Friday May 17th, 2024 (by adding all of the weeks listed below, April 29th is the final week. However, two extra weeks are given in case of any events stopping the schedule.)

For our project, weekends are not needed for work. However, if the team falls behind the number of weeks approximated, weekends will be used as leeway to finish the assigned tasks. Only if they are needed. This schedule does NOT count weekends in "Weeks Approx."
The number of working hours per day is eight. A normal work day would look like this: get to work at 9:00 am and leave around 5:00 pm. If deadlines are approaching fast and taking weekends to work are not enough. Working overtime during the week is required.
Since not all days require everyone in the team to meet, they will work from home on Wednesdays and work at the office the rest of the days: Monday, Tuesday, Thursday, Friday. Again, depending on the group's progress, if needed these days can change.

	Group Task	**Weeks Approx.**	**Notes**

1	***Planning***	1.5 weeks	Getting information together, but not yet implementing it.
2	***Requirements Analysis***	1.5 weeks	All team members meet together to make sure everyone is on the same page before beginning.
3	***Design***	2 weeks	Designing the software and testing the validity of the idea to make sure it will work in real life.
4	***Development***	4 weeks	The longest part of the project because of the coding and it requires a lot of planning.
5	***Reporting (will occur every 2 weeks)***	*Does not count as separate weeks.*	It happens while the project goes on because it involves risk management and control.
6	***Implementation***	3 weeks	This is where the project comes alive. All of the planning and working before this group task is preparing for this.
7	***Quality Assurance (occurs every week after Implementation happens)***	*Does not count as separate weeks.*	Prevents quality failures. Make sure quality is up to date.
8	***Test***	2 weeks	Testing and seeing what to fix from the failures.
9	***Maintenance***	1 week	After testing, maintaining your work is important and finding ways to avoid other errors.
10	***Completion***	1 week	Last few weeks are closing off the project by documenting and

			finalizing everything. Meeting with all stakeholders to see if the project is ready.
11	***Launch***	1 week	The final task, to launch your project to the public.

3.2 Cost, Effort, and Pricing Estimation

The method used to calculate the estimated cost is the Function Point algorithm. The following are the function category counts as shown below: number of user inputs = 16, number of user outputs = 20, number of user queries = 9, number of data files = 13, and number of external interfaces = 7. The processing complexity values for the 14 questions are moderate (2) for PC3, average (3) for PC2, PC5, PC8, PC9, and PC10, significant (4) for PC1, PC4, PC6, PC7, PC12, and PC13, and essential (5) for PC11 and PC14. We are assuming that the productivity of the development team is 6 function points per person-week with a team of 15 people.

> Determine function category count

User Input:

1. Username
2. Password
3. Song ratings
4. Song reviews
5. Discussion posts
6. Discussion comments
7. Liking comments
8. Album ratings
9. Album reviews
10. Artist ratings
11. Artist review
12. Favoriting songs
13. Favoriting albums
14. Favoriting artists
15. Creating playlist
16. Song games input

User Output:

1. Trending music
2. New releases
3. Song recommendations based on user preferences

4. Curated playlist based on user preferences
5. User profile information
6. User's "My Music" page
7. User profile displaying ratings and reviews
8. User profile displaying posts
9. User profile displaying comments
10. Friends/Following page
11. Song details
12. Album details
13. Artist details
14. Discussion board feed
15. Notifications for recommendations
16. Notifications for new music
17. Personalized statistics based on listening habits
18. Playing music
19. Song games
20. Lyrics

User Queries:

1. Find song title
2. Find artist name
3. Find album
4. Find ratings
5. Find reviews
6. Find song games
7. Find friends/users
8. Find playlists
9. Find discussion forums

Data files and Relational tables:

1. User account information
2. User listening data
3. Songs table: by artist, in album, and any other information about a song
4. Albums table: songs in album, by artist, and any other information about the album
5. Artists table: songs, albums, and any other information about the artist
6. Playlists table: created by, songs, and any other information about the playlist
7. Recommended music table
8. Favorites table
9. Playlist table
10. Reviews tables

11. Ratings table
12. Comments table
13. Song games table

External Interfaces:

1. Connecting with streaming platform to play music
2. Connect with youtube for music videos
3. Connect with social media to share and find friends
4. Api for application data
5. Notifications with device or email
6. Mobile application
7. Connect to billboard statistics for trending and new music

> Compute gross function point (GFP)

	Function Category	Count	Complexity			Count x Complexity
			Simple	Average	Complex	
1	# of user input	16	3	4	6	= 16 x 4= 64
2	# of user output	20	4	5	7	= 20 x 4 = 80
3	# of user queries	9	3	4	6	= 9 x 3 = 27
4	# of data files and relational tables	13	7	10	15	= 13 x 10 = 130
5	# of external interfaces	7	5	7	10	= 7 x 10 = 70
					GFP	**= 371**

> Processing complexity (PC)

PC = 0 (no influence), 1 (incidental), 2 (moderate), 3 (average), 4 (significant), 5 (essential)

(1) Does the system require reliable backup and recovery? 4
(2) Are data communications required? 3
(3) Are there distributed processing functions? 2
(4) Is performance critical? 4
(5) Will the system run in an existing, heavily utilized operational environment? 3
(6) Does the system require online data entry? 4
(7) Does the online data entry require the input transaction to be built over multiple screens or operations? 4

(8) Are the master files updated online? 3
(9) Are the inputs, outputs, files, or inquiries complex? 3
(10) Is the internal processing complex? 3
(11) Is the code designed to be reusable? 5
(12) Are conversion and installation included in the design? 4
(13) Is the system designed for multiple installations in different organizations? 4
(14) Is the application designed to facilitate change and ease of use by the user? 5

> Processing complexity adjustment (PCA)

PCA = 0.65 + 0.01(4+3+2+4+3+4+4+4+3+3+3+5+4+4+5) = 0.65 + 0.01(45) = 1.10

> Function point (FP)

FP = GFP x PCA = 371 x 1.10 = 408.10

Estimated Effort:

Assumption: productivity of team is 6 functions points per person-week

E = FP/productivity = 408.10/6 = 68.02 ≈ 68 person – weeks

Project Duration:

Assumption: team has 15 people

D = 68 / 15 = 4.53 ≈ 5 weeks

3.3 Estimated cost of hardware products

Based on the architecture and scale of a project like Noteworthy, hardware needed includes servers, storage, and networking equipment. Due to Noteworthy needing an estimated medium average amount of request volume, the determined amount of application servers would be 12. AWS states at a minimum applications need 5 servers, and at a maximum, they need 16 servers [1]. Abacus states that the cost for a server business with moderate server needs is $1970 [2]. This will be the amount used in the estimation since it's early on. For physical storage, the project will need external hard drives. The average price per gigabyte is $0.014 [3]. The amount of gigabytes needed for physical storage is 1000 as 1 TB of external hard drives for storage in the application is an estimation of what will be needed. The standard hardware, wifi, and construction material for a 24-port network cost is estimated at $5,540 [4]. This amount will be used to estimate networking equipment for the project. With these amounts, the total estimated cost for hardware products is $29.194.

Hardware Expenses				
Items	Description	Units	$ / Unit	Total
Servers	Physical servers	12	$1970	$23640

Storage (GB)	Includes external hard drives	1000	$0.014	$14
Networking Equipment	Includes network switching, wireless activity, and routers	1	$5540	$5540
			Total	**$29,194**

3.4 Estimated cost of software products

To keep an application like Noteworthy up to date and respondent, software needed include cloud services, development tools, licenses for operating systems, database, development tools, libraries, proprietary software, and frameworks. Specific to music streaming, content delivery, and user authentication higher cost allocation will be required. To estimate the cost of these software products the project will use Microsoft Azure for cloud service, Linux as an operating system, MySQL database, Django development framework, Visual Studio Code for IDEs, Python libraries, Microsoft for proprietary software, and Auth0 for user authentication. The products that are free to use and will not be in the table include Linux, Django, Visual Studio Code, and Python Libraries. From Microsoft, the Azure storage pricing is $1545 for 100TB per month [5]. This is the amount that will be used as our storage will be mainly handled in the cloud. Due to using both physical and virtual servers, the project is estimated to need a maximum of 4 servers. The MySQL Standard edition costs $2140 per year [6]. For proprietary software, the project will use Microsoft 365 Business Standard for each project member. The pricing is at $12.50 per user per month [7]. This will be calculated on a yearly basis in the table. For user authentication for a large and scalable platform like Noteworthy, the Auth0 Enterprise plan will be used. The starting price is $30,000 a year [8]. To stream music requires licensing, and to get a variety and large catalog from the BMI business license would cost $5340 per year for a streaming service like Spotify [9]. With all of these factors and pricing, the rough estimate for software expenses yearly is $41,275.

Software Expenses				
Items	Description	Units	$ / Unit	Total
Microsoft Azure	For cloud storage (100TB / month for 1 year)	1	$1545	$1545
MySQL	Licensing for Database	1	$2140	$2140
Microsoft Office	Proprietary software for office productivity	15	$150	$2250
Auth0	User authentication	1	$30000	$30000

Music Streaming licensing	Allowed rights to stream content and music	1	$5340	$5340
			Total	**$41,275**

3.5 Estimated cost of personnel

For the development and maintenance process of building an application, the personnel factors include the number of people to code the end product including their salaries, and training cost after installation. This project's total number of roles is estimated to be about 15 people. The amount for each role is listed below. Using Glassdoor, the estimated salaries for different roles in the United States can be derived: Project Manager is $159k, Business Analyst is $99k, UI/UX Designer is $93k, Architect is $221k, Database Manager is $102k, Developers is $112k, and QA Testers is $77k [10]. The amounts in the table refer to monthly payments. The average training cost per employee is $1252 [11]. This creates a rough estimate, due to team size being subject to change and location dependencies, of $158,260 per month for personnel expenses.

Personnel Expenses				
Items	Description	Units	$ / Unit	Total
Team Member	Role: Project Manager	1	$13250	$13250
Team Member	Role: Business Analyst	2	$8250	$16500
Team Member	Role: UI/UX Designer	1	$7750	$7750
Team Member	Role: Architect	1	$18420	$18420
Team Member	Role: Database Manager	1	$8500	$8500
Team Member	Role: Developers	6	$9300	$55800
Team Member	Role: QA testers	3	$6420	$19260
Training costs	To keep employees up to date on skills	15	$1252	$18780
			Total	**$158,260**

4. Test Plan

This test plan outlines the procedures for testing the unit get_top.py, which is responsible for retrieving the top n artists and songs from a .csv file containing the top tracks streamed on

Spotify in 2023. In the testing process, the goal is to verify that the software unit accurately retrieves the top n artists and songs, ensuring that the fetched data matches the expected results from the .csv file. Items that are in the scope of this test include retrieval of top n artists and songs, precision of data, and proper handling of errors and edge cases. However, the performance and efficiency testing of the database itself and this querying process is considered to be out of scope. The get_top function will serve as the test item, and its parameters "n" and "category" are the features to be tested. We will use the black-box testing method to mirror the perspective of the user as closely as possible, without having any knowledge of the inner workings of the function. We will use the unittest and pandas.testing frameworks in Python to perform the unit testing. There are no specific hardware requirements (standard hardware is acceptable), but the required software includes the aforementioned frameworks and their dependencies, as well as a simulation of the database (a truncated .csv file is sufficient).

Test Cases:

1. **Retrieval of top 5 songs**
    a. Test data- a pandas dataframe with 5 rows containing the expected output
    b. Expected result- success
2. **Retrieval of top 5 songs**
    a. Test data- the same dataframe from test case 1 but with a different song and artist name
    b. Expected result- failure (the 4th (song, artist) pair should be observed to be different between the two dataframes)
3. **Retrieval of top 10 songs**
    a. Test data- a pandas dataframe with 10 rows containing the expected output
    b. Expected result- success
4. **Retrieval of top 5 songs**
    a. Test data- the same dataframe from test case 3 but with a different song and artist name
    b. Expected result- failure (the 1st (song, artist) pair should be observed to be different between the two dataframes)
5. **Retrieval of top 3 artists**
    a. Test data- a pandas dataframe with 3 rows containing the expected output
    b. Expected result- success
6. **Retrieval of top 3 artists**
    a. Test data- the same dataframe from test case 5 but with two sets of differing artist names
    b. Expected result- failure (the first two rows of artist names should be observed to be unequal)
7. **Retrieval of number 1 artist**
    a. Test data- a pandas dataframe with 1 row containing the expected output
    b. Expected result- success
    c. Expected result- success
8. **Retrieval of number 1 artist**
    a. Test data- the same dataframe from test case 7 but with a different number of streams for the artist

b. Expected result- failure (the top artist should match but their number of streams should be different than expected)

The testing of this specific unit will be conducted over a period of 1 day, and the test cases will be executed in any random sequence since that should not impact the outcome. The test deliverables are the test results and logs, as well as any bug reports or issues identified during testing.

5. Comparison with similar designs

Our app references the following applications and websites:

- **Spotify:** a digital music platform that allows users to access millions of songs.
    - **Key features:**
        - Spotify Wrapped: at the end of each year, Spotify provides a wrap-up for users. Spotify Wrapped's theme changes every year but the key information that Spotify includes every year is: top 5 artists, top 5 genres, top song, top genre, top podcast, and minutes listened.
        - Generated playlists: Spotify generates playlists based on the user's music taste. Some notable playlists include: a playlist that updates depending on the time of day; playlists based on an artist's genre or genre in general (i.e. a pop playlist that includes pop songs from different artists).
        - Personality games: time-to-time, Spotify will release a small music based game. Usually, these games serve to find out what type of person you are–they're basically personality games but based on the user's musical choices.
- **Receiptify:** a website that allows users to connect to their Spotify account and displays their 10 most-played tracks from the last month, the last 6 months, most listened to genre(s), and all time in a receipt-like form.
    - **Key features:**
        - Top-songs based on time period: same as the description above: users can figure out their 10 most-played tracks from the last month and the last 6 months.
- **Letterboxd:** a website that allows users to discover new films and share their love for films. Similar to Rotten Tomatoes.
    - **Key features:**
        - Reviews: users are allowed to write reviews for each film. They can rate them with stars and write a few comments about their opinions. Users are also able to like and comment on reviews. Additionally, reviews are sorted by popular and most recent.
        - Playlists for movies: users are able to create playlists for movies and make comments on other users' playlists.
        - News: Letterboxd provides news articles for each movie.

- **Genius:** a website that helps users gain knowledge about songs and their lyrics
    - **Key features:**
        - Song annotations: for each song, users can highlight lyrics and add their interpretation of it. Additionally, sometimes artists can add verified interpretations of lyrics.
- **Other inspirations:** Goodreads, Storygraph, Rotten Tomatoes.
    - For conciseness, the key features in these three applications are similar to the key features listed in Letterboxd. Both Goodreads and Storygraph are platforms that allow users to rate books whereas Rotten Tomatoes is for films as well.

How Noteworthy Stands Out

We noticed that there aren't many platforms that focus on sharing and reviewing music. So, our app incorporates the key features from Letterboxd, Receiptify, Spotify, and Genius into one place with additional features that these platforms do not provide. Here are the key features of our app:

- **Song critique:** for each song, users can add comments to lyrics. These comments can be reactions (i.e. "This is an amazing word choice!!") or it can be a critique. Additionally, users are able to react as well as reply to other comments as well for a specific lyric.
- **Song games:** there are many song games out there that are fun to play with friends. Our app would include various song games such as karaoke, guess the lyric, finish the lyric, etc. For each, the songs will be randomized but be based on the user's data or whatever is trending.
- **Personalization:** our app focuses on creating an optimal, personalized experience for our user. We have a section in the user profile dedicated to providing personalized songs, genres, artists, and playlists. Additionally, our app provides statistics on a weekly, monthly, and yearly basis. These statistics would range from top song(s), artist(s), album(s), etc to a personality based statistic (i.e. what type of aura do they have).
- **Community:** in addition to being able to add comments to lyrics, our app incorporates different ways to build community within the musical world. Our app focuses on building artist to fan, fan to fan, critic to fan, and critic to artist connections by incorporating a discussion section. This section allows critics, fans, and artists to post announcements, opinions, etc and allows other users to comment on them. Our app takes into account music critics as users which applications like Spotify and Letterboxd doesn't include. Additionally, the discussion section acts as a way to get news and updates about the musical world (i.e. when an artist releases an album). Similarly, users can also share music to other users.

6. Conclusion

Through a group effort, Noteworthy has transformed from a simple brainstorm into a fully planned software project. Although we have not done any implementation (coding) for the project, based on the work completed the process of implementation has been made easier. All of the work that has been done through both individual and team efforts has been deliberately

completed with the goal of moving the project into a more complete version. Although the original proposal came early in the process, the changes that have been made have been very minimal. Most of any changes don't affect the original ideology and instead expand on it to increase the functionality of Noteworthy. In the process of defining the various diagrams and requirements, there has been an increase in cohesiveness. These changes and expansions came as the project moved through the various processes with the goal of strengthening the project. Overall, the Noteworthy software system combines various individuals' efforts to make a functioning plan and background for a software should we wish to continue with implementation.

7. References

[1] AWS OpsWorks User Guide, "Best Practices: Optimizing the Number of Application Servers, " *Amazon Web Services,* 2023. [Online]. Available: https://docs.aws.amazon.com/opsworks/latest/userguide/best-practices-autoscale.html. [Accessed Nov. 11, 2023].

[2] Abacus, "How Much Does a Server Cost for a Small Business?," *Abacus*. [Online]. Available: https://goabacus.com/how-much-does-a-server-cost-for-a-small-business/. [Accessed Nov. 11, 2023].

[3] Andy Klein, "Hard Drive Cost Per Gigabyte," *Backblaze*, November 29, 2022. [Online]. https://www.backblaze.com/blog/hard-drive-cost-per-gigabyte/#:~:text=Overall%2C%20the%20drop%20in%20the,cost%20per%20terabyte%20since%202009. [Accessed Nov. 13, 2023]

[4] Thomas Kinsinger, "How much does a small network setup cost in 2023?" *Encomputers*, 2023. [Online]. https://www.encomputers.com/2023/03/small-network-setup-cost/. [Accessed Nov. 12, 2023].

[5] Azure, "Azure Blob Storage pricing," *Azure*, 2023. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/storage/blobs/. [Accessed Nov. 14, 2023].

[6] Redress Compliance, "Mysql License – A Complete Guide To Licensing," Redress Compliance, June 21, 2023. [Online]. https://redresscompliance.com/mysql-license-a-complete-guide-to-licensing/. [Accessed Nov. 10 2023].

[7] Microsoft, "Find the best Microsoft 365 plan for your business," Microsoft. [Online]. https://www.microsoft.com/en-us/microsoft-365/business/compare-all-microsoft-365-business-products. [Accessed Nov. 15, 2023].

[8] Joel Coutinho, "Auth0 Pricing: The Complete Guide [2022]," Super Tokens, Nov. 15, 2022. [Online]. https://supertokens.com/blog/auth0-pricing-the-complete-guide. [Accessed Nov. 13, 2023].

[9] RetroCube, "How Much Does It Cost To Create A Music Streaming App Like Spotify?" RetroCube, Sept. 22 2021. [Online]. https://www.retrocube.com/blog/how-much-does-it-cost-to-create-a-music-streaming-app-like-spotify/. [Accessed Nov. 16, 2023].

[10] Glassdoor, “Salaries,” *Glassdoor*. [Online]. https://www.glassdoor.com/Salaries/index.htm. [Accessed Nov. 16, 2023].
[11] Rachel Blakely-Gray, “What Is the Cost of Training Employees?,” *Patriot*, Aug. 30, 2022. [Online]. https://www.patriotsoftware.com/blog/payroll/cost-training-employees-average/. [Accessed Nov. 15, 2023].

8. Presentation Slides

https://docs.google.com/presentation/d/1ivIfMhJMj0Y4kNQTYdi60y6SrmD3oJbs8bVQYTeds5E/edit?usp=sharing