Vaultify

"Vaulting music memories."

Sprint 4 Deliverables

Software Design and Documentation
1:30 PM - 3:35 PM Section
Professor John Sturman

The Band:

Matthew Bui, Michael Lam, Dillon Li, Michelle Li, Thomas Orifici

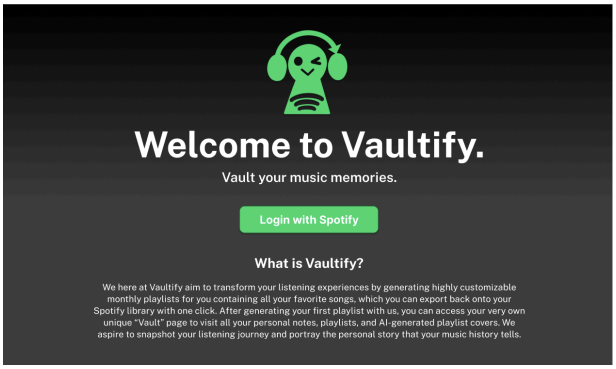
28 June 2024

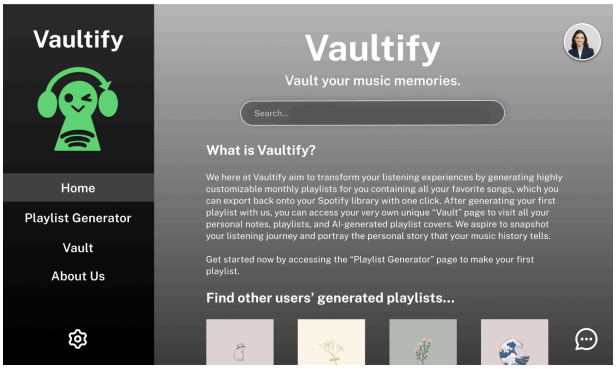
VaultifyInquiries@gmail.com

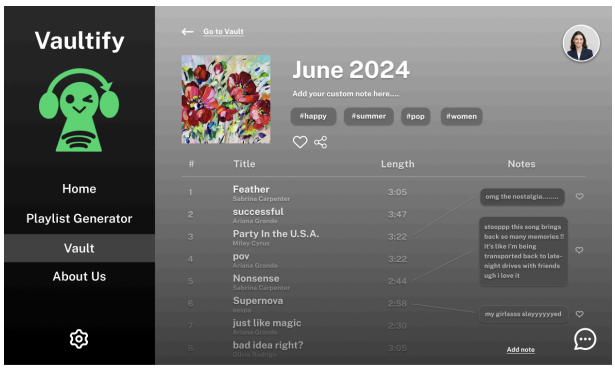
Product Interface Mockup

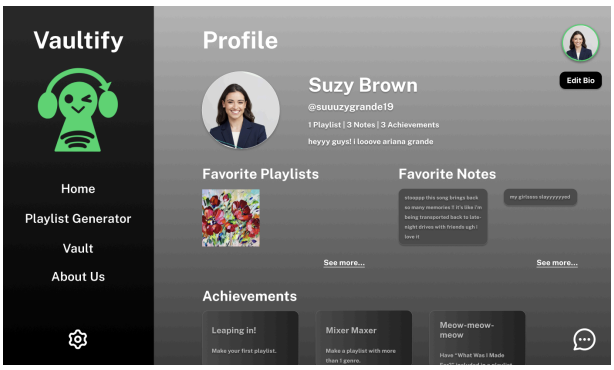
For the product interface mockup, the Vaultify team uses Figma to prototype our mockups. This ensures that we create a clear and precise design that is close to our vision for the final application while also paying close attention to its color identity and overall user experience. We then organized these prototypes using Google Slides for a structured presentation. Finally, using a guided and interactive format, we recorded and uploaded the presentation to YouTube.

1. This is the link to our Figma mockups: https://www.figma.com/design/XG1n9RN9kNbRaHKTxj7Wru/Vaultify?node-id=0-1&t=1H1cfO7R0W2IB9MI-1.
2. This is the link to our Google Slides presentation: https://docs.google.com/presentation/d/1aDVouqVqc7M6ctk9L367rHsWSmlSq-PvatksMAwfrLU/edit?usp=sharing.
3. This is the link to our mockup recording:
4. https://youtu.be/7yDKI9BIG5E.









User Stories

Vault	
Story:	As a nostalgic person, I want a place that displays my music listening habits each month so that I can look back on my memories with music.
Points:	40
Conversation:	We can incorporate a “Vault” page that stores the total generated playlists that a user has created with the website. It can store specific dates that the playlists have been generated and direct users to all of their playlists.
Confirmation:	We will test to make sure that playlists are visible in their “Vault” page after both manual and automatic generation. We will test that the date when the playlist is added is consistent with its placement in the vault. We will ensure that playlist titles and covers are displayed properly upon change.

Spotify Integration	
Story:	As a Spotify user, I want my playlists to be automatically transferred to my Spotify account so I can listen on my preferred platform.
Points:	13
Conversation:	We can include a button that redirects the user to their playlist in Spotify after playlist generation. We can automatically add the generated playlists to the user’s Spotify library.
Confirmation:	We will test that the button takes the user directly to the correct playlist on Spotify. We will test that playlists are added to the user’s library after both manual and automatic generation. We will test that Spotify and website playlists contain the same content.

Playlist View	
Story:	As a songwriter, I want to be able to view and write notes on songs in my playlists so that I can keep track of my inspirations over time.
Points:	20
Conversation:	We can add a “Playlist View” page that displays information on a particular playlist. We can add an annotation feature that allows users to write notes on certain songs that have things to say about. The “Playlist View” page can be accessed through the “Vault” page.
Confirmation:	We will test that all songs in a playlist are properly displayed on the page. We will test that annotations are stored properly and will reappear when the page is revisited. We will test that the annotations are pointing to the correct song upon any movement on the screen.

Share	
Story:	As an influencer, I want a “Share” button with my top monthly songs so that I can share my habits with my followers.
Points:	8
Conversation:	We can add an infographic which contains easy to share information such as mood of songs, frequently visited songs, etc. We could also add a scannable QR code leading to the user’s profile on our site.
Confirmation:	We will test that the generated infographic contains statistics accurate to the user. We will ensure that the QR code redirects directly to the user’s profile.

Achievements	
Story:	As a gamer, I want to be able to gamify my music-listening experiences so that I feel rewarded while using the app.
Points:	20
Conversation:	We can have the user's "Profile" page display achievements for milestones such as how many months a user has been using the platform or for their favorite genre. We can store this information in the database.
Confirmation:	We will ensure that completing a task always gives the user the correct achievement. We will test that ending a usage streak works properly.

Custom Playlist Covers and Titles	
Story:	As an aesthetics enthusiast, I want to be able to generate cover images for my playlists and edit the titles so that my music library stays cute.
Points:	40
Conversation:	We can allow users to either select an image to set as their playlist cover, or use an AI model to create a cover for the playlist using the mood of the playlist as a template. We can allow users to input custom titles.
Confirmation:	We will ensure that when users select a playlist cover or title, the title and the cover remain the same every time you revisit the playlist. We will test cover generation with several moods and themes to ensure that the images accurately portray them.

Automatic Playlist Generation	
Story:	As a lazy person, I want playlists to automatically be generated and added to my Spotify account so that I don't have to manually check the app every month.
Points:	40
Conversation:	We can set up a date library that automatically generates playlists on the first of every month. Users would have to opt-in to automatic generation.
Confirmation:	We will test that playlists are automatically generated and appear on both the site and the user's Spotify library. We will ensure that this only occurs for opted-in users. We will make sure the timing of playlist generation is correct.

Playlist Generation Customization	
Story:	As a professional full-time nerd, I want to be able to customize my playlist generation so that each playlist is perfectly tailored to my preferences.
Points:	13
Conversation:	We can allow users to specify different attributes they want their playlist to have, such as number of songs, playlist length, and generation interval (i.e., weekly, monthly, seasonal, etc). We could store this information in the database for each user.
Confirmation:	We will need to test and ensure that the playlists actually adhere to these constraints. We will make sure that every combination of playlist generation actually works and that nothing crosses over with other options. For example, users must specify playlist length or number of songs. We will make sure that custom generation time frames work properly and override the default 1 month timeframe.

Email Reminders	
Story:	As a forgetful and busy person, I want to be reminded when a playlist is generated so that I can keep track of my listening history.
Points:	13
Conversation:	We can set up a notification system that would email a user when the past month's playlist has been generated for them. The emails could link to the generated playlist. These notifications could be implemented using Firebase Cloud Messaging or Twilio SendGrid.
Confirmation:	We will make sure that emails are sent only when a new playlist is generated. We will ensure that the notifications are going to the right email for each user.

Welcome Page	
Story:	As an old person, I want to be given an overview of the platform so that I don't get confused.
Points:	3
Conversation:	We can include a "Welcome" page as the landing page for the site, detailing the purpose and functions of the platform. This page can outline each key feature of our application.
Confirmation:	We will make sure the instructions we give are accurate, and are updated every time we make any significant changes to the platform.

Supplemental Specification

Must

1. The application will require users to log in via Spotify OAuth. Their profile information will be saved in Vaultify.
2. The application will allow users to generate a playlist with their top songs for the month.
3. The application will feature a “Vault” page where users can view their playlists and personal notes generated by the app. This page will feature both a timeline and list view.
4. The application will allow users to view the songs of each of their playlists in the vault by clicking on each playlist.
5. The application will allow users to annotate their playlists and songs with personal notes and memories, reminiscent of a diary.
6. The application will allow users to import their own playlist covers from their local system.
7. The application will have the option for email notifications when new playlists are generated.
8. The application will include a settings page that will allow users to toggle between dark and light modem, logout, and change preferences regarding monthly automatic playlist generation and email notifications.

Should

1. The application will run smoothly on a mobile interface.
2. The application will allow users to customize their playlist generation with options such as playlist length, number of songs, and generation time interval.
3. The application will provide a playlist sharing function.
4. The application will determine the mood of the user’s playlist based on the genre of the songs in the playlist. This will be displayed in the
5. The application will provide the option for AI-generated playlist covers based on the top mood playlists.
6. The application will allow users to sort their playlists on the “Vault” page in different ways by using a filter (i.e. by time created, by genre tags, by amount of notes).
7. The application will include achievements for activities such as discovering a new genre or using the app for consecutive months.

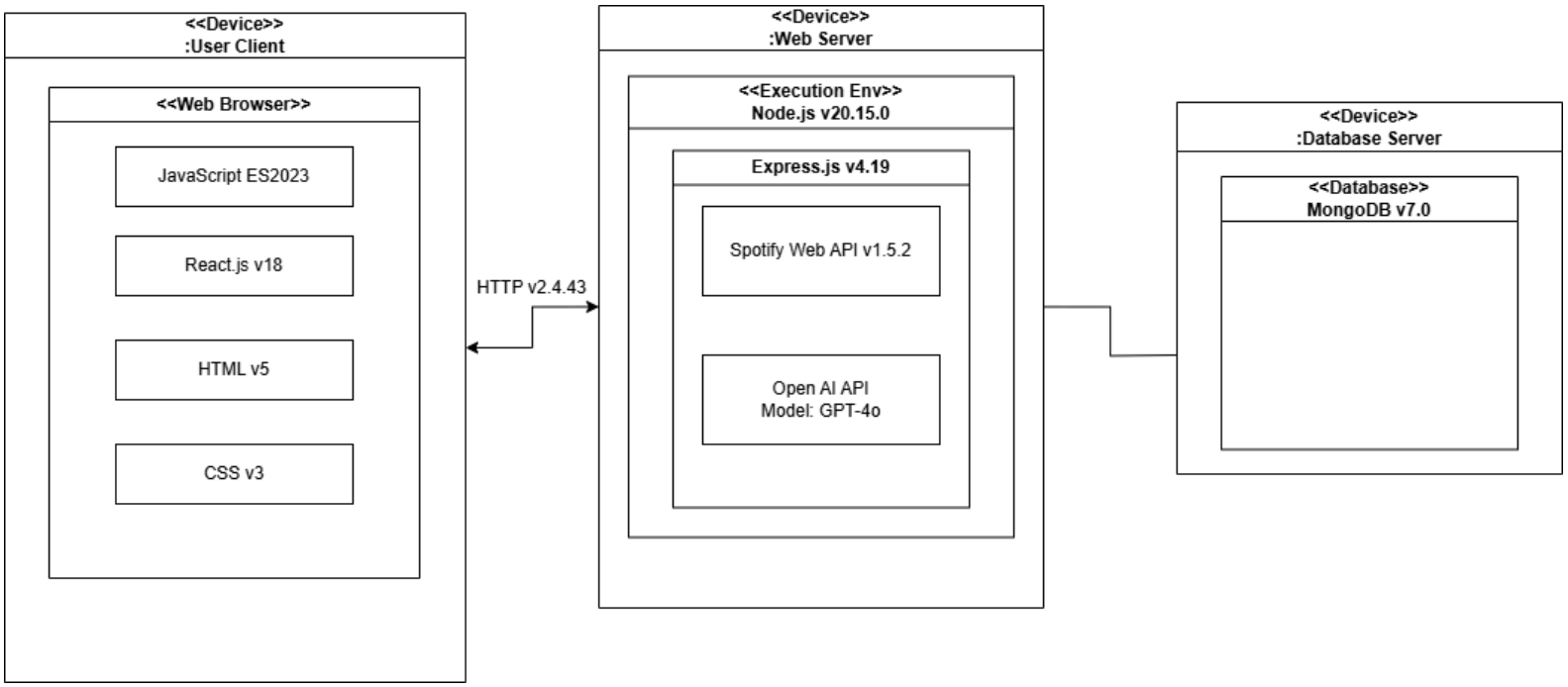
Could

1. The application will allow users to share playlists directly to social media platforms.
2. The application will suggest new songs based on the user's listening history.
3. The application will allow users to organize playlists into folders or categories within the "Vault" page.
4. The application will provide detailed analytics on the user's music preferences including insights into their favorite artists and genres of the month.
5. The application will allow users to set up notifications for new song releases and updates from their favorite artists.

Won't

1. The application won't have a chat box for user-to-user interactions.
2. The application won't include advertisements.
3. The application won't have a built-in music player.
4. The application won't support offline mode for playlist generation.
5. The application won't provide direct integration with other music streaming services besides Spotify.
6. The application will not have an alternative login method to Spotify OAuth.

Deployment Diagram



Use Cases

SpotifyLogin	
Identifier:	UC0
Description:	The SpotifyLogin use case models a user logging in via Spotify OAuth through the Vaultify website.
Actors:	The user (Music Listeners/Spotify Users)
Precondition:	1. User has started the Vaultify application
Flow of events:	1. Use Case starts when user selects the login with spotify option. 2. If user has Spotify login saved on their system: a. System redirects user to Spotify OAuth, which lists the permissions Vaultify will need to use. b. If user selects “I agree option”: i. System redirects user to the “Home” section of the application. c. If user selects “cancel” option: i. System redirects user back to the “Welcome” section of the application. 3. Else: a. System redirects user to Spotify login. b. User inputs their Spotify login information. c. System redirects user to Spotify OAuth, which lists the permissions Vaultify will need to use. d. If user selects “I agree option”: i. System redirects user to the “Home” section of the application. e. If user selects “cancel” option: i. System redirects user back to the “Welcome” section of the application.
Postcondition:	The system has redirected the user to Spotify OAuth, where the user makes a decision on the permissions they need to grant us. If they choose to accept, they will be redirected to the “Home” page where they can access the navigation bar. Otherwise, they will be redirected back to the “Welcome” page.

Navigation	
Identifier:	UC1
Description:	The Navigation use case models a user choosing a feature of the application to interact with.
Actors:	The user (Music Listeners/Spotify Users)
Precondition:	1. User has logged in via Spotify's OAuth.
Flow of events:	1. Use case starts when the user views the list of available features on the site. 2. The user selects the feature they would like to interact with out of those available 3. The system redirects the user to the portion of the application containing that feature.
Postcondition:	The user can interact with their selected feature and navigate easily throughout the Vaultify site.

AnnotatePlaylist	
Identifier:	UC2
Description:	The AnnotatePlaylist use case models a user annotating a playlist through the Vaultify website
Actors:	The user (Music Listeners/Spotify Users)
Precondition:	1. User has logged in via Spotify's OAuth. 2. User is viewing a playlist within their "Vault" (listening history).
Flow of events:	1. Use case starts when the user views teh songs of a playlist that's been generated. 2. User selects selects a song and chooses the "add note" option. 3. System prompts the user to input a note for the specified song. 4. User inputs the note and selects the "save" option. 5. System redirects the user back to their playlist view. Their note is now visible and coincides with the selected song.
Postcondition:	The user's playlist view shows the note that they added.

GeneratePlaylist	
Identifier:	UC3
Description:	The GeneratePlaylist use case models a user generating a playlist through the Vaultify website.
Actors:	The user (Music Listeners/Spotify Users)
Precondition:	1. User has logged in via Spotify's OAuth. 2. User has navigated to the "playlist generation" section of the application
Flow of events:	1. Use Case starts when user selects the "Playlist Generator" tab in the navbar. 2. User chooses whether to generate playlist based on number of songs or length 3. If length selected: a. System prompts the user to choose a playlist length. b. User chooses between 1 hour, 2 hours, or 5 hours, or a custom timeframe. 4. Else: a. System prompts the user to select a number of songs. b. User chooses between 25, 50, 100, or a custom number of songs. 5. The User chooses whether they would like to enable monthly playlist generation. 6. If monthly generation is selected: a. The system prompts user to choose if they'd like to receive email notifications b. The user selects either yes or no. 7. User selects "generate playlist" option 8. System redirects the user to a view of their playlist.
Postcondition:	User is viewing their generated playlist within the application.

CheckAchievements	
Identifier:	UC4
Description:	TODO
Actors:	The user (Music Listeners/Spotify Users)
Precondition:	1. User has logged in via Spotify's OAuth. 2. User is viewing a playlist within their "Vault" (listening history).
Flow of events:	1. Use case starts when the user views teh songs of a playlist that's been generated. 2. User selects selects a song and chooses the "add note" option. 3. System prompts the user to input a note for the specified song. 4. User inputs the note and selects the "save" option. 5. System redirects the user back to their playlist view. Their note is now visible and coincides with the selected song.
Postcondition:	The user's playlist view shows the note that they added.

AutomaticGeneration	
Identifier:	UC5
Description:	The AutomaticGeneration use case models a user having a playlist automatically generated for them after a month.
Actors:	The user (Music Listeners/Spotify Users)
Precondition:	1. User has logged in through Spotify's OAuth. 2. User has generated a playlist and selected the "monthly generation" and "email notifications" options.
Flow of events:	1. Use case starts when the first of a new month has arrived. 2. System sends an email to the user saying that a new playlist has been generated and providing a means to redirect to it in the "Vaultify" application 3. User chooses the "redirect" option. 4. The "Vaultify" application is opened and the user is automatically brought to the view section for their new playlist.
Postcondition:	The user is viewing their generated playlist in the application.

SharePlaylist	
Identifier:	UC6
Description:	The SharePlaylist use case models a user generating a shareable image summarizing one of their playlists.
Actors:	The user (Music Listeners/Spotify Users)
Precondition:	1. User has logged in via Spotify's OAuth. 2. User is viewing a playlist within their "Vault" (listening history).
Flow of events:	1. Use case starts when the user selects the "share" option while viewing a playlist 2. System displays an image depicting the top songs and mood of the playlist along with a QR code. Also prompts the user to download the image. 3. User selects the "download image" option 4. System downloads the playlist image to the user's device.
Postcondition:	The user has downloaded a shareable image summarizing and linking to their playlist.

FindPlaylist	
Identifier:	UC7
Description:	The FindPlaylist use case models a user navigating to a particular playlist in their "vault".
Actors:	The user (Music Listeners/Spotify Users)
Precondition:	1. User has logged in via Spotify's OAuth.
Flow of events:	1. Use case starts when the user selects the "vault" section of the application to navigate to. 2. System redirects the user to their "vault". Playlists are displayed in chronological order and show the dates they were created. 3. User selects a playlist to view. 4. System redirects the user to the view for that playlist.
Postcondition:	The user is viewing their selected playlist.

Work Breakdown Structure

Coding

1. Implement the design for the “Welcome” page
    1.1. Add a login button that redirects the user to Spotify OAuth.
    1.2. Add a splash screen with branding and loading animation.
    1.3. Add a general blurb about what Vaultify is.
2. Implement the design for the “Home” page
    2.1. Add a general blurb about what Vaultify is.
    2.2. Add a concise user tutorial.
    2.3. Display other users’ recently created playlists.
3. Add a navigation bar on the left side of the site.
    3.1. Add Vaultify’s name and logo that redirects users to the “Home” page when clicked.
    3.2. Add clickable links for all site pages.
    3.3. Add functionality such that clicking on a link or a button redirects to that page or feature.
4. Implement playlist generation backend functionality.
    4.1. Authenticate with Spotify using OAuth to obtain access tokens.
    4.2. Configure API endpoints for accessing user data.
    4.3. Fetch top tracks based on user listening data and generate a playlist with those songs.
    4.4. Utilize Date.js library to configure automatic playlist generation.
5. Implement the design for the “Playlist Generator” page
    5.1. Create a form for users to input playlist details (date, length, cover theme, etc.).
    5.2. Add customization options for playlist features..
    5.3. Allow users to upload or generate an AI playlist cover.
    5.4. Implement functionality to export playlists and redirect users directly to Spotify.
    5.5. Enable sharing of playlists via link and image generation.
6. Implement the design for the “Vault” page.
    6.1. Implement a timeline display that shows playlist titles and covers in chronological order from top to bottom. Include top notes from each playlist.
    6.2. Implement a list display that displays each playlist in smaller, easier to navigate boxes.
    6.3. Configure such that clicking a playlist cover redirects to that playlist’s page.
7. Implement the design for the “Playlist View” page.
    7.1. Display title, cover, and all songs for the playlist.
    7.2. Add the ability for users to add notes to their songs.
8. Implement the design for the “About Us” page:
    8.1. Display the team’s mission and FAQs, as well as the team members’ descriptions.
    8.2. Ensure that the page is informative and visually appealing.

9. Implement the design for the “Settings” page:
    9.1. Add options for editing personal information, including user bio and profile picture.
    9.2. Implement light/dark mode toggle functionality.
    9.3. Add settings for notification preferences and automated playlist generation.
    9.4. Implement logout functionality.
10. Implement the design for the “Profile” page:
    10.1. Create info box for general user information, including name, username, email, and profile picture.
    10.2. Create info box for achievements.
    10.3. Create info box for favorite playlists preview.
    10.4. Create info box for favorite notes preview.
    10.5. Add “See more...” buttons for both playlist and notes so that when clicked, displays all playlists and notes respectively.
    10.6. Implement functionality to edit personal info such as user bio.
11. Install needed dependencies and libraries (i.e., Bootstrap, Vite, Axios, cookie-parser, cors, react-dom, react-router-dom)
    11.1. Set up “concurrently” dependency to ensure that running application locally will be simple for testing
12. Setup MongoDB database
    12.1. Ensure that all team members have set up MongoDB Atlas and MongoDB Compass tools.
    12.2. Store fields such as Spotify token, previously generated playlists, user settings, and notes.
    12.3. Implement retrieval of relevant data for display on the site.
13. Train AI model for AI-Generated playlist covers
    13.1. Collect and preprocess user listening history from Spotify. Analyze the top mood of songs.
    13.2. Integrate Open AI API into project
    13.3. Query AI to generate a playlist cover based on the top mood of songs.
14. Deploy application onto Vercel and/or Heroku
    14.1. Deploy the application to both Vercel and Heroku for staging and testing.
    14.2. Assess the pros and cons of each platform using factors such as deployment speed, scalability, and cost.
    14.3. Make a team decision on which platform to use based on the evaluation.

Interface and Design

1. Select a consistent set of design guidelines such as color schemes, typography, and button styles.
    1.1. Ensure all mock-ups adhere to the design system for a cohesive look and feel.
2. Create a mock-up for the “Welcome” page.
    2.1. Design the layout and visuals to create a strong first impression.
    2.2. Include elements such as the application name, logo, Spotify login button, and introductory text.
3. Create a mock-up for the “Home” page.
    3.1. Design the navigation bar and layout for easy access to different sections.
    3.2. Include sections for other users’ playlists, FAQ, and a quick user tutorial.
4. Create a mock-up for the “Playlist Generator” page.
    4.1. Integrate fields for playlist customization.
5. Create a mock-up for the “Vault” page.
    5.1. Design the timeline view for displaying all playlists and notes.
    5.2. Design a list view for easier readability.
6. Create a mock-up for the “About Us” page.
7. Create a mock-up for the “Settings” page.
8. Create a mock-up for the “Profile” page.
    8.1. Design the layout for displaying user information, achievements, and favorite items (i.e., favorite notes and favorite playlists).
9. Create interactive prototypes for each mock-up to demonstrate user flows and interactions.
10. Ensure Accessibility Considerations
    10.1. Ensure all mock-ups follow accessibility guidelines, including color contrast, font size, and keyboard navigation.
    10.2. Design with accessibility in mind to cater to users with different needs.

Documentation

1. Update Project Schedule
2. Update the README.md file with new code, features, and usage instructions
3. Update version notes, including new features, improvements, and bug fixes.
4. Work on Interim Release deliverables
    - 4.1. Record testing results for all implemented features to ensure they meet the required standards.
    - 4.2. Document all bugs discovered during testing along with the steps taken to fix them.
5. Work on Beta Release deliverables
    - 5.1. Prepare and distribute a survey to beta users for feedback.
    - 5.2. Refine the user interface based on beta user feedback
    - 5.3. Record interviews with volunteer users to gather qualitative feedback.
    - 5.4. Document user feedback to identify areas for improvement.
    - 5.5. Update the list of bugs and fixes based on user testing and feedback.
6. Complete Final Deliverables
    - 6.1. Complete final testing of Vaultify to ensure all functionalities are working as intended.
    - 6.2. Ensure all bugs from interim and beta releases are fixed.
    - 6.3. Perform a security audit to ensure the application is secure.
    - 6.4. Conduct performance testing to ensure the application meets performance standards.
    - 6.5. Prepare a final installation and deployment guide on GitHub.
    - 6.6. Update all documentation to reflect the final state of the application.
7. Complete Final Presentation
    - 7.1. Create a slides presentation covering the project's application objectives, development process, features, testing results, user feedback, and future plans.
    - 7.2. Include visual aids such as screenshots, charts, and graphs to enhance the presentation.
    - 7.3. Prepare a live demo of the application to showcase its functionality and user experience.
    - 7.4. Highlight the major milestones and accomplishments throughout the project.

Testing, Verification, Validation

1. Set up testing environment
    - 1.1. Ensure that the development environment is configured correctly for all team members.

1.2. Verify that the necessary API keys and authentication tokens are available for all team members.
1.3. Set up any required mock data or test users in Spotify.

2. Prepare tools and frameworks
   2.1. Set up testing frameworks like Postman and automated testing tools like Jest for Node.js.
   2.2. Ensure that all dependencies and libraries are installed.
3. Test Spotify API calls
4. Test user login through Spotify OAuth
   4.1. Verify that access tokens are correctly issued and stored securely.
5. Test “Home” page
   5.1. Test navigation bar: Logo refreshes or redirects to “Home” page
   5.2. Test navigation bar: Redirects to “Playlist Generator” page
   5.3. Test navigation bar: Redirects to “Vault” page
   5.4. Test navigation bar: Redirects to “About Us” page
   5.5. Test user profile: Redirects to “Profile” page
   5.6. Test display of most recent playlist
6. Test “Playlist Generator” page
   6.1. Test playlist generator: Create a playlist
   6.2. Test playlist generator: Customize playlist features
   6.3. Test playlist generator: Add notes to playlist
   6.4. Test playlist generator: Create playlist cover
   6.5. Test playlist generator: Export playlist and redirect user to Spotify
   6.6. Test playlist generator: Share playlist via link
7. Test “Vault” page
   7.1. Test timeline: Display all playlists and notes
   7.2. Test timeline: Edit selected playlist and notes
   7.3. Test timeline: Add playlists
   7.4. Test timeline: Share playlists via link
8. Test “About Us” page
   8.1. Ensure the display of product’s mission, description, features, and a user tutorial.
9. Test “Settings” page:
   9.1. Test logout functionality.
   9.2. Verify user email addresses.
   9.3. Test automated email reminders and notifications.
   9.4. Test light/dark mode functionality.

10. Test “Profile” page:
    10.1. Edit personal information such as user bio.
    10.2. Display all unlocked and locked achievements.
    10.3. Ensure that hidden achievements are embedded in the code but not displayed in the profile page until unlocked by the user.

10.4. Display favorite notes and playlists.
10.5. Ensure that a pop-up displays all favorite notes when "See more notes..." is selected.
10.6. Ensure that a pop-up displays all favorite playlists when "See more playlists" is selected.

11. Performance Testing
    11.1. Test a high load by making multiple requests to the endpoint at the same time.
    11.2. Measure the response time for each request.
    11.3. Ensure that the response times are within acceptable limits and that the API does not crash under load.
12. Security Testing
    12.1. Test for common security vulnerabilities such as SQL injection, XSS, and CSRF.
    12.2. Ensure that the API only returns data to authenticated and authorized users.
    12.3. Validate that sensitive information is not exposed in error messages or logs.

Meetings, Presentation

1. Schedule regular team meetings and small group meetings to discuss progress, challenges, and next steps.
2. Conduct user feedback meetings to discuss findings from user testing and interviews.
3. Complete Final Presentation
    3.1. Discuss the feedback given by beta users and how it was incorporated into the development process.
    3.2. Ensure that all team members are comfortable with presenting and understanding the contents of the slide.
    3.3. Ensure that there is proper transition between speakers,
    3.4. Gather and include testimonials from users and stakeholders.
    3.5. Explain the future plans and potential enhancements for the project.
    3.6. Prepare answers to potential questions from the audience.
    3.7. Ensure that the presentation is engaging and interactive, with opportunities for Q&A.

Updated Project Schedule

Sprint Number	Tasks
Sprint 1 6/4 @ 1:00 PM - 6/11 @ 11:00 AM	The team will complete our Sprint 1 Deliverables, including their vision statement, user scenarios, and schedule. Additionally, they will create a GitHub Repository with a skeleton React project and confirm that all team members' development environments are working. They will also establish coding standards for the project.
Sprint 2 6/11 @ 1:00 PM - 6/18 @ 11:00 AM	The team will create a welcome page with the project name, logo, description, and a "Login with Spotify" button. This button will allow users to log in to their Spotify accounts using the Spotify web API. They will also create a "home" page that displays user information and settings.
Sprint 3 6/18 @ 1:00 PM - 6/25 @ 11:00 AM	The team will complete their Sprint 4 Deliverables, including their user stories, product interface mockup, supplemental specification, deployment diagram, use cases, work breakdown structure, and updated project schedule. They will also work on implementing an "About Us" page and create a mock up for the playlist generation page.
Sprint 4 6/25 @ 1:00 PM - 7/2 @ 11:00 AM	The team will create a playlist generation page displaying a "Generate Playlist" button. They will also use the Spotify API to collect the user's top 50 songs for the past month, generate a playlist for the user with these songs, and add the playlist to the user's library.
Sprint 5 7/2 @ 1:00 PM - 7/9 @ 11:00 AM	The team will create a "Playlist Successfully Generated" page, with the playlist name, the playlist cover, and a success message. Clicking on the playlist name or cover will redirect the user to the playlist on Spotify. They will also create a database to store user email addresses and Spotify authorization codes. They will add UI to the playlist generation page to allow users to opt-in to monthly emails and playlist generation.
Sprint 6 7/9 @ 1:00 PM - 7/16 @ 11:00 AM	The team will configure the app so that playlists automatically generate each month for opted-in users. They will also integrate email notifications for opted-in users. Additionally, They will complete the documentation for the interim release, including their design approach, sequence diagrams, CRC cards, static class diagram, and mock up interface test plan.

Sprint 7 7/16 @ 1:00 PM - 7/23 @ 11:00 AM	The team will create a “Vault” page where users can view, add notes to, and interact with their generated playlists. This will include a playlist view that displays all songs and notes for a playlist. They will develop two views for the vault page, a simple view and a timeline view.
Sprint 8 7/23 @ 1:00 PM - 7/30 @ 11:00 AM	The team will integrate custom timeframes for playlist generation (weekly, biweekly, monthly, seasonal, yearly). They will also allow users to choose a custom playlist length (10 songs, 25 songs, 50 songs, 100 songs) and add a “New Songs Only” option for playlist generation, where only songs that are new to your top songs are added to the playlist. They will add these options into the UI of the playlist generation page.
Sprint 9 7/30 @ 1:00 PM - 8/6 @ 11:00 AM	The team will integrate AI playlist cover generation with multiple theme options based on the month and the songs that are in the playlist. They will also implement mood analysis for generated playlists. Additionally, they will complete the testing documentation for the beta release.
Sprint 10 8/6 @ 1:00 PM - 8/13 @ 11:00 AM	The team will add a “Share” feature that allows users to share their listening habits with their friends. They will also add achievements that users can unlock by using the app for consecutive months. In addition, they will fully test the product and prepare documentation for the final release.

Interim Release (7/19)

The interim release of Vaultify will contain the platform’s base features. The user will be able to log in with their Spotify account, generate a playlist with their top 50 songs for the past month, and view that playlist in Spotify. Additionally, they will be able to opt in for automatic monthly playlist generation and email notifications. The UI will include a “Welcome” page, “Home” page, “Playlist Generation” page, “Playlist Successfully Generated” page, and “About Us” page.

Beta Release (8/9)

The beta release of Vaultify will add additional features to the interim release such as new options for playlist generation (“New tracks only,” song count, and length), the “Vault” page where users can view and add notes to their generated playlists, playlist cover generation through mood analysis of songs, and a share feature. The UI will be updated to account for these features as well as for feedback from the interim release.

Project Status Report

Since Sprint 1, the Band's development of Vaultify has proceeded in an efficient and organized manner. The team has extensively adhered to the schedule outlined in our project schedule, and after the completion of helpful and valuable documentation that we discussed and began in class such as a set of user stories and use cases, a deployment diagram, site mockups, software requirements, a work breakdown structure, and CRC cards, the team feels much more confident in our ability to proceed with the technical aspects of the project, like programming and deploying the web application. In particular, the development of our mockups, first done out by sketch and later refined using Figma, has solidified the layout and execution of the platform's interface design and features. All members of the team have also been researching MongoDB and the Spotify Web API and are ready to implement these technologies in our Vaultify app. Additionally, the team has solidified the structure of the project by outlining the classes that we're planning to use for the project, which better prepares us for implementation.

Our current risks involve ensuring that all the members of the team learn and familiarize themselves with the technical aspects of our project, such as creating our playlist generation algorithm, AI playlist cover generation algorithm, as well as our Vaultify Chatbot's AI response algorithm. On top of that, some team members that have not had a coding task yet will need to ensure that they are better familiarized with JavaScript, HTML, CSS, React, Node.js, and Express.js by this point, or at the latest over the break, so that the pace of the project moves along as intended and implementation of the project can begin.

Moving forward, after a lot of planning, designing of class diagrams and structures, writing up and submitting our Sprint 4 Deliverables, and stringent consideration of the features we wanted to prioritize, the team's next steps are to dive into the actual coding portion of the project. First, we will implement the frontend for the "Welcome," "Home," and "About Us" pages using guidance from the Figma mockups. Then, we will move on to implementing the playlist generation algorithm, database integration, and monthly automation. As of now, the team is unanimously looking to prioritize the playlist generation feature first for the upcoming interim release on July 19. The team is excited to move into the next stage of development in their project and get a first working and fully functional version of Vaultify up and running.

Contribution Summary

Matthew Bui:
Matthew added to and fine tuned the status report so that it also included risks and other details. Matthew also contributed to the supplemental specifications section and helped to brainstorm the project's user stories, use cases, and deployment diagram.

Michael Lam:
Michael wrote down the conversation and the confirmation for the user stories. He also helped redraw the deployment diagram from class. Additionally, he helped write and revise the use cases to help fit the desired format. Lastly, he helped with the recording of the product interface mockup video, assisting Michelle with the interface walkthrough.

Dillon Li:
Dillon contributed to the documentation of the team's work breakdown structure and Vaultify's supplemental specifications. He also wrote out the expectations of Vaultify in its interim and beta releases and contributed to the development of user stories.

Michelle Li:
Michelle completed the product interface mockups on Figma, recorded the mockup video, and wrote out the Vaultify application's supplemental specifications. She also contributed to developing the technical details of the team's work breakdown structure and editing the project status report.

Thomas Orifici:
Thomas updated the project schedule to account for changing deadlines. He also worked on the status report and formatted the team's user stories. Additionally, Thomas fledged out the project's use case and deployment diagram.