COMPSCI 732 Project Proposal

Dandelion Dolphins

March 19, 2023

Introduction

In today's digital age, music and video streaming services have become an integral part of our daily lives. With the advent of technology, we can access a vast library of music and videos with just a few clicks. However, there is always room for innovation and improvement. This is where our proposed webapp comes in.

Our goal is to develop a novel music and video streaming service interface using APIs such as Spotify. The proposed webapp, named "Party Playlist", will provide users with the ability to search and add songs to a shared playlist, even if they don't have a Spotify account. The playlist will be managed by a host who can curate user additions as per their preference.

The motivation behind developing this webapp is to provide a platform where users can come together and create a collaborative music experience. It's not always easy for everyone to have a Spotify account, but with "Party Playlist", anyone can join in and contribute to the playlist, creating a fun and inclusive atmosphere for all.

In the following proposal, we will discuss the details of the webapp, its features and implementation details. We believe that "Party Playlist" has the potential to become a popular and successful app, and we are excited to share our vision with you.

Related work

There exist many web apps and streaming services that offer similar features and functionality to music streaming services. For example, Spotify, YouTube Music, and Apple Music are all popular music streaming services.

- The things work well
    1. These web apps allow users to stream millions of songs over the Internet. Users can listen to the songs anytime.

2. Users can classify the songs by their own interests. Put songs into their favorite playlist and share them.
3. These web apps can present the lyric when users cannot clearly hear the songs.
4. Users can search for some singers. The web apps will provide the introduction and some representative works about them.
5. The songs can be downloaded and cached so that users can listen to them without a network.

- Something can be improved

  1. Provide a platform for users to share songs, playlists, and comments with each other.
  2. Provide users with more personalized recommendations, and make more accurate recommendations based on users' listening history and preferences.
  3. Enable users to intelligently recommend some soothing music to users by searching keywords, such as soothing music.
  4. When the user has the melody of a song but does not know the song's name, the app can recognize the song's name by receiving the song played by the user. Even the app can recognize it by humming a few paragraphs.

Requirements

- Must-Haves

  1. User authentication: Enable secure sign up and login functionality for users, enabling them to access their own playlists and interact with other users.
  2. Search functionality: Users should be able to search the Spotify catalog for songs, artists, albums, and playlist names.
  3. Real-time updates: When a user adds or removes a song from the playlist, all other users should see the changes in real-time without refreshing the page.
  4. Responsive design: Ensure that the webapp works well on different devices, including desktops, laptops, tablets, and smartphones.
  5. Host curation: The host should have the ability to curate the shared playlist by removing or reordering songs added by other users.

- Should-Haves
  1. Playlist curation: Allow the host of the party playlist to curate user additions, so that inappropriate or duplicate songs can be removed.
  2. Social sharing: Allow users to share their playlists on social media or with other users via a link.
  3. Multi-platform integration: Enable users to discover and add songs from various music services, including Apple Music and Net-cloud music.
- Could-Haves
  1. Song recommendations: Provide personalized song recommendations for users based on their listening history or preferences.
  2. Playlist voting: Allow users to vote on which songs should be played next.
  3. Comments: Allow users to chat with eachto discuss the music selection or other topics.
- Nice-to-Haves
  1. Song recommendations Plus: Enhance the recommendation system by integrating the AI servise to enable fuzzy search functionality for songs using keywords beyond song titles, like ”sad songs” or ”driving songs”.
  2. Song lyrics: Display the lyrics of the currently playing song, which may be either user-generated or uploaded by users themselves.
  3. Dark mode: Users could toggle between a light and dark theme for the webapp interface.

Technologies

Based on the project requirements, we will use the following technologies and materials to complete the music player web application:

- React is a popular front-end JavaScript library for building user interfaces, which will be used to create the user interface for the web application.
- The MERN stack is an acronym for MongoDB, Express, React, and Node.js. These technologies will be used to build the full-stack web application. MongoDB will serve as the database, Express will be used to create the server-side API, and Node.js will be used as the runtime environment for the server-side code.

- The Spotify API will be used to retrieve music data and user information from Spotify. The API provides endpoints for authentication, search, playback, and more.

- Element UI is a Vue.js UI framework that provides pre-built UI components based on the Google Material Design system. It will be used to enhance the overall user experience of the application. Since Element UI is designed for Vue.js, it will require some customization to work with React. However, this will provide an opportunity to learn and apply new skills, and to integrate Vue.js components into a React application.

- Redux is a state management library for JavaScript applications. It will be used to manage the state of the application, such as user authentication status and music playback information.

- SCSS and LESS are both CSS preprocessors that allow developers to write CSS in a more organized and efficient way. They provide features such as variables, nesting, and mixins that make it easier to create and maintain CSS stylesheets. One of them will be used to style the application and provide a more consistent and maintainable CSS codebase. The choice between SCSS and LESS will be based on personal preference and previous experience.

- Scrum is an agile project management methodology that emphasizes teamwork, collaboration, and iterative development. It will be used to manage the project and ensure that it is completed on time and within budget.

- Git is a version control system that allows multiple developers to work on the same codebase. GitHub is a web-based platform that provides hosting for Git repositories. They will be used to manage the source code for the application and collaborate with other developers.

In addition to the above technologies, independent learning and research will be conducted to explore other libraries, frameworks, and technologies that can be used to enhance the functionality and user experience of the application. This may include technologies such as Webpack for module bundling, Jest for testing, and ESLint for code quality analysis.

Project Management

- Agile Strategy and Sprint Planning: We will use an agile strategy for the development of the app, with a focus on iterative development and continuous feedback. We will aim to have 4-6 sprints in total, with each sprint lasting for 1-2 weeks, depending on the complexity of the features being developed.

Here's a potential sprint breakdown:

  - Sprint 1 (Week 1-2): Set up project infrastructure, including creating a project plan, defining user stories, creating a basic UI, and setting up Git repositories.
  - Sprint 2 (Week 3-4): Implement basic functionality to allow users to search and add songs to a playlist, including integration with a music streaming API such as Spotify.
  - Sprint 3 (Week 5-6): Implement basic user authentication and allow hosts to create and share a playlist.
  - Sprint 4 (Week 7-8): Implement user profiles, social sharing, and playlist curation features.
  - Sprint 5 (Week 9-10): Implement user voting, fine-tune the UI, and carry out testing.
  - Sprint 6 (Week 11-12): Finalize any outstanding features, carry out further testing, and prepare for deployment.

- Our team will implement the following communication and coordination strategies:

  1. Product Backlog: We will maintain a product backlog which includes all current ideas for the app. We will aim to complete at least one sprint (1-3 weeks) worth of work from the product backlog.
  2. Sprint Planning Meeting: We will hold a sprint planning meeting at the beginning of each sprint to decide on the content and scope of the sprint. During this meeting, we will choose items from the product backlog to work on, and define the sprint goal and sprint backlog (divided into "to-do", "doing", and "done" categories).
  3. Daily Scrum: We will hold daily stand-up meetings (also known as "Daily Scrums") where each team member will report on their progress, any challenges they are facing, and what they plan to work on next. The focus of these meetings will be on sprint development.
  4. Sprint Review: At the end of each sprint, we will hold a sprint review meeting to demonstrate the completed work to stakeholders and get feedback.
  5. Sprint Retrospective: Also at the end of each sprint, we will hold a sprint retrospective meeting to discuss what went well, what didn't go well, and what we can improve in the next sprint.

By implementing these strategies, we aim to ensure effective communication and collaboration among team members, as well as to keep the project on track and within schedule.

- Version Control Strategy:

    1. Our team will use Git as the version control system for the project, with a central repository hosted on Github.
    2. Each team member will work on their own branch and submit pull requests for code review and merging into the main branch.
    3. We will implement automated testing and continuous integration using a tool such as Travis CI or CircleCI to ensure the quality of the codebase.

Risks in team project

Possible risks

- Personnel risk

    1. Lack of development experience and basic development foundation (language, experience, tools, etc.).
    2. Team members are not interested in their own projects and have low enthusiasm for development, so they cannot meet the required product functions and performance requirements.

- Development environment risks

    1. Development tools are not ready.
    2. Development tools are not as effective as expected, and developers need more time to get familiar with and learn the use of development tools.
    3. Different development environments may encounter some incompatible problems due to different systems. For example, macOS is not compatible with the interface and functions implemented by Windows, and there are subtle differences.

- Design and implementation risks

    1. The design is too simple to develop corresponding functions.
    2. The design is too complicated, leading to some useless work and affecting efficiency.
    3. The design is poor and needs to be redone.

How to avoid these risks

- Avoid personnel risks

  1. Choose the language that most people in the team are familiar with to develop, try to ensure that everyone has a certain foundation, and then let everyone develop in their own areas.
  2. When the team determines the project, let everyone express their own ideas as much as possible, put forward suggestions and ideas for the project, and improve the development enthusiasm of each team member.

- Avoid development environment risks

  1. Identify and unify development tools in advance.
  2. In the early development process, if it is found that the development tool is not useful and cannot reach the desired goal, communicate with other team members in time and make adjustments.
  3. Determine the development environment, and understand the differences between different systems and the effects of project development in advance. If the computer system is different in the team, it can be modified slightly at the time of final integration to ensure that the minority is subordinate to the majority.

- Avoid design and implementation risks

  1. The goal of the design should not be too simple or too complex. Before the design, the technical ability of each member in the team should be understood and a reasonable design plan should be formulated according to the technical level of the team.
  2. In the design process, weekly meeting summary should be made in time, team schedule should be made, to ensure that each stage is carried out according to the plan, and the quality of each stage should be controlled.