PROJECT PROPOSAL BY JOEY HUANG

SampleHub

Project Abstract

Music producers often spend hours digging through multiple websites and videos to find the right loops and samples. This project aims to simplify this process with SampleHub, a web application that gathers audio content and metadata from different sample libraries and organizes them into a single platform. Unlike other single source websites, SampleHub gathers music from multiple sources, retrieves metadata like BPM, key, and genre, and allows users to quickly preview and download sounds. Users will also be able to save and create custom sample packs of their favorite sounds. By reducing time spent searching and encouraging the discovery of new sounds, SampleHub provides value to artists and producers while expanding access to a variety of resources.

Conceptual Design

SampleHub Stack:

1. The Frontend (React) provides UI for search, preview, and favorites.
2. The Backend (Node.js + Express) handles requests, scraping/APIs, normalizes metadata, manages user data, and serves audio preview links.
3. A Scraper/API system gathers data from sample sites or calls official APIs (FreeSound API, Youtube Data API). Scraping is only performed where allowed and otherwise use APIs.
4. A database (MySQL) holds metadata, user accounts, favorites, and saved packs.
5. Libraries for audio analysis like music-metadata or soundtouch are needed.

Background

The purpose of SampleHub is to simplify the process of looking for loops, samples, or inspiring audio material. It does this by gathering loops and samples from multiple free/public sources and combining them on one searchable web platform. It will also provide a “sampling discovery” section that features random YouTube videos which users can explore for sampling ideas. Users will be able to save favorites, build custom sample packs, and even upload their own original sounds to share or use later. SampleHub will only gather data from sites that allow scraping or sites that provide APIs.

Existing platforms offer parts of this functionality, but with limitations. For example, Splice(https://splice.com/) provides a large royalty-free sample library, but much of its content is gated by subscription or limited by credits. Platforms like Looperman(https://www.looperman.com/) offer free loops and samples uploaded by users under royalty-free terms, but it is focused mostly on static user uploads and doesn’t provide discovery of YouTube sources. SampleHub differs from both by combining multiple sample sources, YouTube discovery, and allowing users to upload their own sounds all on one platform. It emphasizes open access and free content along with the relevant metadata (BPM, key, genre) for each loop or sample.

Proof of Concept

https://github.com/jyyhuang/SampleHub-POC

Required Resources

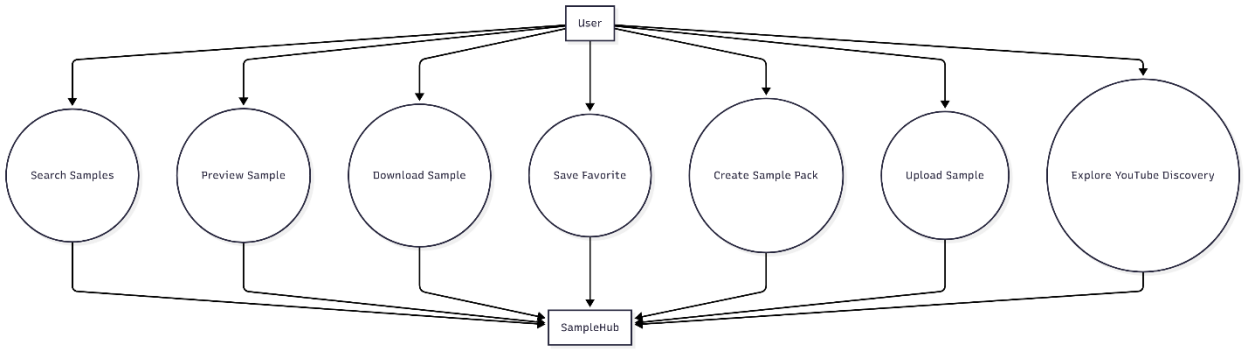
- Development laptops/desktops with text editor/IDE with Node.js installed.
- Required packages and tools can be installed on own development environments.
- Knowledge of web scraping basics and metadata handling for audio files.

- Access to free sample libraries and public YouTube videos.

Project Timeline

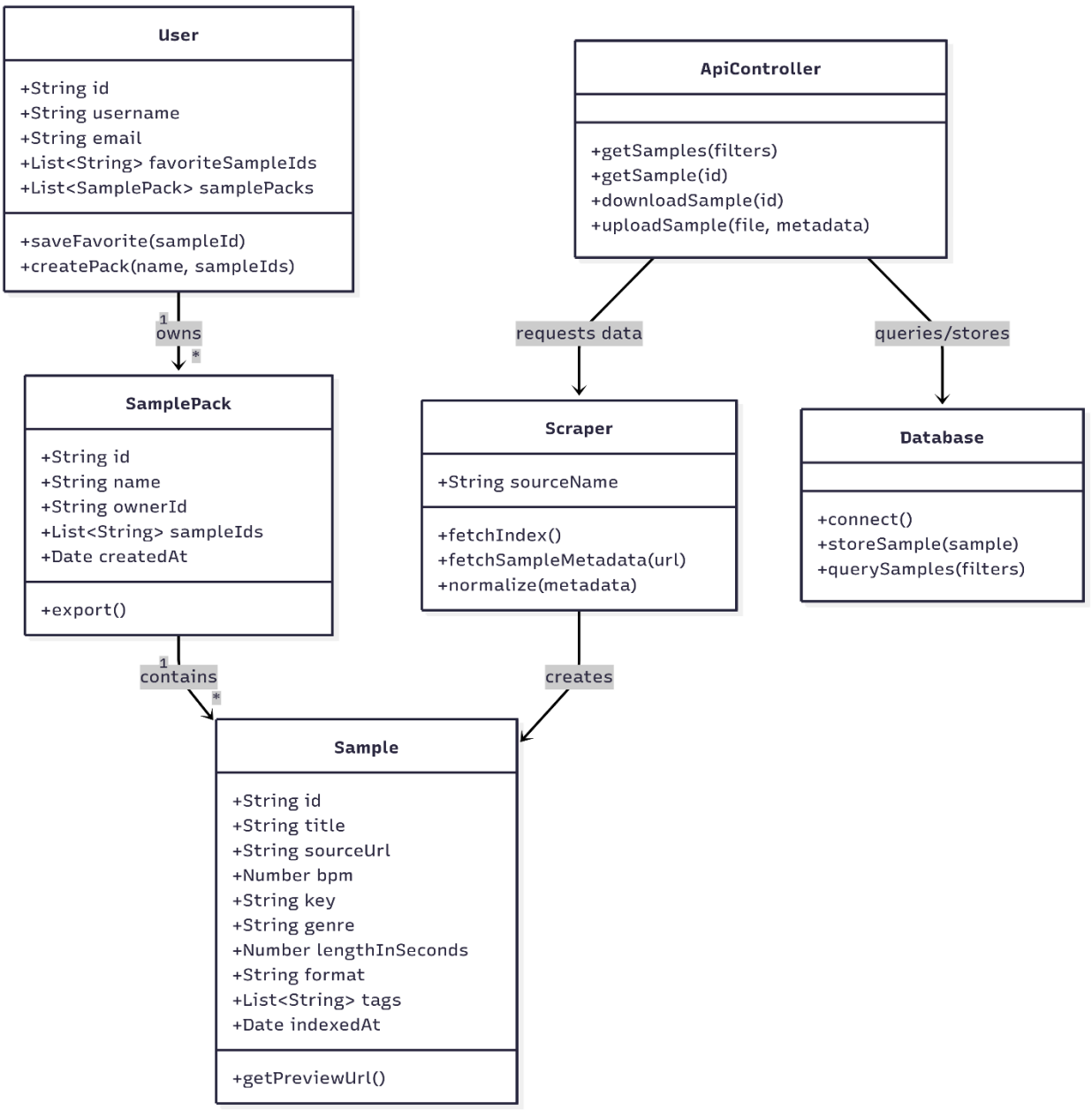
- Week 1-2: Set up backend and frontend. Build/test scripts against one sample source.
- Week 3-4: Integrate database for storing metadata and user favorites. Implement search and filtering.
- Week 5-6: Build features: preview audio, custom sample packs, saving favorites.
- Week 7: Add "sampling discovery" feature.
- Week 8: Testing, bug fixes, documentation.

Use Case



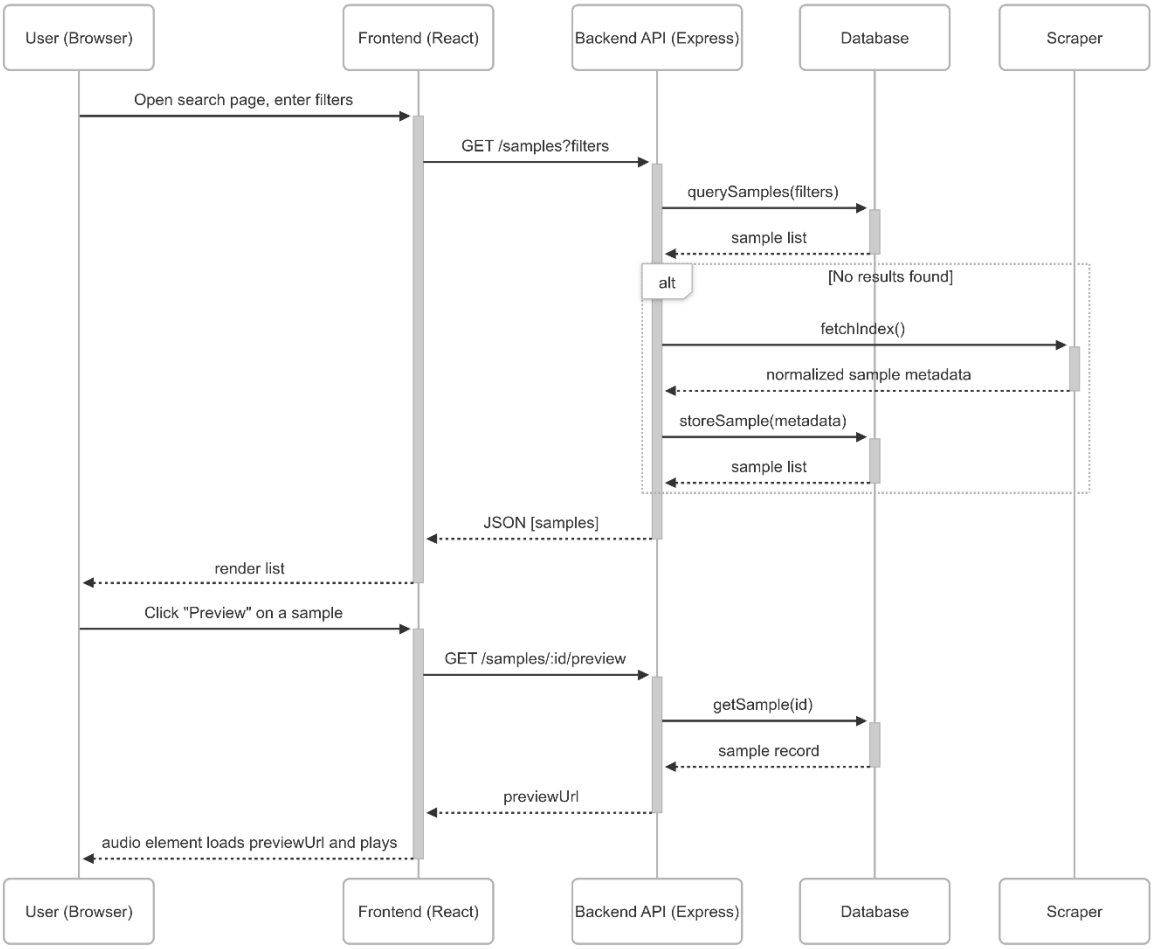
- **Search Samples:** User enters filters and results are returned.
- **Preview Sample:** Play in-browser preview of audio.
- **Download Sample:** Download the audio file.
- **Save Favorite/Create Sample Pack:** Features for persistent selections.
- **Explore YouTube Discovery:** Random YouTube videos are given for sampling ideas.

Class Diagram



- **Sample:** Represents a single audio item. It stores metadata, the original source URL, and its id. The method getPreviewUrl() returns a playable URL.
- **User:** Stores user info, favorites, and owned packs. The saveFavorite(sampleId) method takes a sample's id and saves that sample as a user's favorite. The createPack(name, sampleIds) creates a sample pack with "name" and list of sample ids.
- **SamplePack:** A grouping of Samples. It has its own id and a reference to the owner. It is exportable.
- **Scraper:** Source-specific adapter that fetches and normalizes metadata.
- **Database:** Class for the database. Persistence layer (CRUD)
- **ApiController:** Api endpoints for frontend.

Sequence Diagram



- When the user makes a search request in the React frontend, it sends a GET /samples request to the backend API.
- In the backend, the API queries the database for matching samples and returns if they exist. If results are missing, the backend calls the Scraper to fetch data from external sources.
- The Scraper normalizes metadata and saves it into the database.
- The backend sends a JSON response containing the matching samples and the frontend renders the results in the UI.
- When a user clicks “Preview,” the frontend requests /samples/:id/preview. The backend gets the sample record from the database and returns a preview URL.
- The front end loads the preview URL into an HTML <audio> element, allowing the user to play the sample directly in the browser.