There isn't much different here from your first draft, so my concerns about all the backend work remain. I would have liked to have seen more detail on the front end.

# Podplay.me

## The Multiplatform Podcast Player

G 4
F 6 – not enough on front end
C 6
U 3
I 2 – no discussion at all of front-end issues
S 4
A 4
P 6

35

Ian McGaunn & David Zimmelman

# Contents

# Project Goal

The goal of the project is to create a web application for aggregating and streaming podcasts. This application will allow users to customize their experience and take their preferences with them across devices. It is also a goal of the project to implement the application using modern web technologies including the WebAudio API on the client-side, and Node.js and MongoDB on the server-side.

# Features

## Simple Podcast Feed Management

Podplay is at its core a podcast manager. This means that being able to subscribe to, search for and make playlists of podcasts are all central features. Podplay will leverage the breadth of the iTunes podcast catalog via the iTunes search API to help users find their favorite feeds and organize them.

## Synchronization of User Preferences

As users subscribe to feeds and create playlists using Podplay, their settings will be saved to their accounts and will be restored upon logging in. This will allow users to keep their settings across all of their devices.

## Clean User Experience

Good

The screen mockup in Figure 1 shows the layout for Podplay. Podplay is split into two main components: the library view and the player view. The largest part of the user interface is the library view, which houses the main functionality of Podplay. The library view presents a user's saved podcasts and provides a search function to find new podcasts. To simplify the user interface, additional features (for example, browsing by genre) are accessible through the collapsible navigation menu; this approach is modeled after techniques used in mobile applications. The player view provides the ability to customize the current playlist and stream podcast episodes.

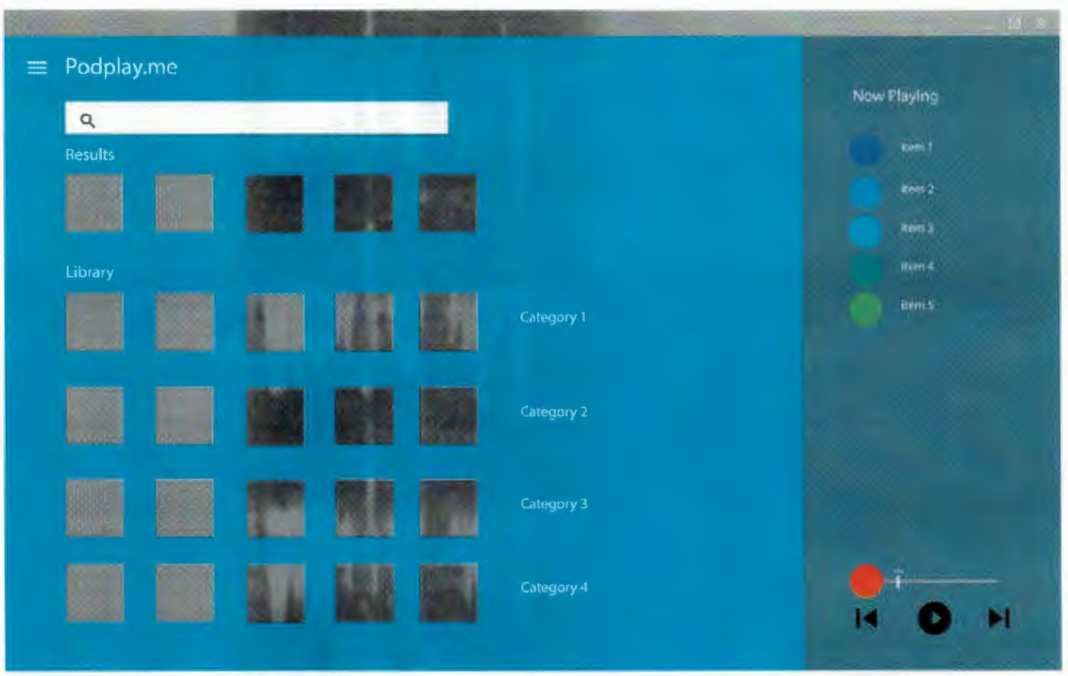


*Figure 1: Basic design concept*

Good

## Native Browser Podcast Player

Podplay will support podcast playback using native browser APIs rather than relying on third-party middlewares such as Flash or Java. Reliance on W3C

approved specifications, will improve performance and expand the number of devices that Podplay can run on without additional dependencies.

## User Descriptions

Podplay is designed for avid podcast listeners who would prefer a web application that allows them to preserve their settings across devices over a native mobile app. Podcast listeners of ~~any~~ either gender and any age can find use for Podplay, as it is a simple way to organize and listen to podcasts from any platform with a modern web browser. For users already committed to a mobile podcast app, Podplay is a good companion for when their mobile phone is not available or when they are spending a prolonged period of time at a computer where a web application might be more strategic.

dangerous, red flag words
Any age? Not my 3-month old granddaughter!

## Component Details

Podplay is primarily implemented as a Node.js server-side application which exposes a minimal API for searching and listing podcast feeds to the client-side web pages. The client will use standard HTML5/CSS to present the user with a clean UI.

## Templating

Views in Podplay.me are rendered server-side with Jade templates, which greatly reduces the amount of markup that has to be written and provides a method of code reuse for our web page structure. Requests for these views are handled server-side by Express, a Node.js web framework that simplifies routing and templating.

## iTunes API Integration

The main view of Podplay is focused around the podcast feeds a user has subscribed to. To enable the user to locate their preferred feeds and to help the application aggregate information, the iTunes search API will be used. Access to the entire iTunes catalog of podcasts is a feature that many mobile apps have neglected to provide.

## Data Persistence

A MongoDB database will store users' account information and individual preferences, and also cache information about commonly accessed podcast feeds to circumvent exceeding iTunes API rate limits.

## Browser Audio

The actual audio player for Podcasts will be implemented with the HTML5 <audio> element and the WebAudio API. The WebAudio API provides a simple way to decode and play audio in the browser, making it a great candidate for a pure HTML5 audio player. Since these standards are implemented in most

modern web browsers (Firefox, Chrome and Safari), this approach should be straightforward to implement in a browser agnostic manner.

## Potential Issues

### iTunes API Limitations

The API provided by iTunes is not very flexible, nor is it very robust. The first major issue is the possible rate limits on the API requests. Although Apple does not explicitly define any such limitations, it has been reported that they exist when exceeding roughly 10,000 requests per day. If this problem arises, a caching system will be implemented into the project so that any request made will first hit the server's cached podcast directory for a "quick result" and only make the hard request to Apple's API when necessary (e.g., a deep search).

The second issue with the iTunes API is the inability to retrieve a full list of all available podcasts, or even to provide a page offset for a search. This means the maximum number of results that can be retrieved for any request is 200. Unless the previously-mentioned caching system is implemented, the project will be limited to the inferior search request of the iTunes API. However, this seems to be the standard used by similar podcasting applications, so at the very least, this project will be on-par to its competitors.

### Non-Uniform Data Formats

Because podcast feeds are individually managed, it is possible to run into non-standard data formats within individual feeds. Episode-specific information, such as summaries, could possibly have oddly-named keys, or might not even exist at all. In these (presumedly rare) situations, the data will not be shown to the users (who will likely see a message similar to "not available" where the information would normally be), until a special rule can be added in the server to find and parse the non-uniform data.

What about GUI issues ?!?

## Project Timeline

| Milestone | Expected Date | Dave | Ian |
|---|---|---|---|
| **Week 1**: Development environment setup and bare-bones server running with dedicated domain. | **15 Feb 15** | X | X |
| **Week 2**: Node server has partial iTunes API implementation. | **22 Feb 15** | X | |
| **Week 3 (Alpha 1)**: Extremely basic web client for testing server implementation. Ability to search, view and stream podcasts. | **26 Feb 15** | | X |

| | | | |
|---|---|---|---|
| **Week 3**: Node server application has full iTunes implementation. | **1 Mar 15** | X | |
| **Weeks 4-5 (Alpha 2)**: Fully styled client website. Implements all server functionality: searching and viewing podcast data as well as HTML audio streaming of individual podcasts. | **15 Mar 15** | | X |
| **Week 6**: Partial user account system implemented in server. Database integration, secure transmission methods, and user data model. Provides methods for registering, signing in and signing out. | **22 Mar 15** | X | |
| **Week 7**: Full user account system implemented in server. Provides methods for updating user preferences, subscriptions, and playlists. | **29 Mar 15** | X | |
| **Weeks 8-9 (Alpha 3)**: Client side implementation and GUI polishing for user accounts. | **12 Apr 15** | | X |
| **Week 10 (Beta 1)**: Add any extra needed polish to application and resolve any open issues/bugs. Usability test for Alpha 3. | **14 Apr 15** | X | X |
| **Weeks 11-12 (Release)**: Time permitting, work on additional backlogged features for final release. | **1 May 15** | X | X |

# Acceptability Criteria

## Required Features

- A functional player for streaming podcasts
- A UI that allows users to subscribe to podcast feeds and create playlists.
- Searching using the iTunes API
- Server-side storage of user preferences to allow users to maintain their settings across multiple devices.

## Extra Features

- Server-side caching of podcast feed information to circumvent iTunes API rate limits.
- Notify users when new items are pushed to podcast feeds.