Project: *Extraordinary Rhymes*

Developer(s): James Yen

Software Development Plan

(To be included in the SDF as section 4.)

4.1 Plan Introduction

This Software Development Plan provides the details of the development for *Extraordinary Rhymes*, a client-side web app which aids the user in writing song lyrics.

4.1.1 Project Deliverables

The final Software Development File will include:

1. A Project Proposal & Justification, delivered on Week 2.
2. A Requirements Specification Document, delivered on Week 5.
3. A Software Development Plan, delivered on Week 9.
4. Status Reports, delivered every other week, starting on Week 9.

4.2 Project Resources

A breakdown of resources used in the development of the project.

4.2.1 Hardware Resources

* Eee PC 1015PE
  + Small netbook that I use for development on campus.
* Windows 7 desktop computer.
  + PC that I built for myself. If I develop at home I use this computer.

4.2.2 Software Resources

* Sublime Text 3.
  + For easy editing.
* Notepad++.
  + Also for easy editing.
* GitHub Desktop.
  + For version tracking.
* Git
  + For version tracking.
* Chromium
  + For testing the app.
* Google Chrome
  + For testing the app.
* Iceweasel
  + For testing the app.
* Vim
  + For writing code.

4.3 Project Organization

App consists of:

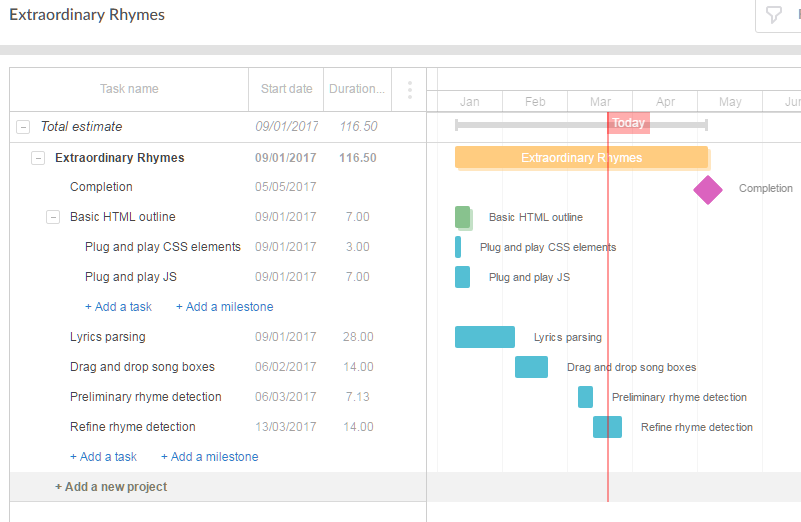
* JavaScript files, they handle all the DOM and business logic.
  + There is JavaScript that rips the lyrics from the page and applies some regex stuff in order to strip punctuation.
  + There is JavaScript that works with the API via synchronous web requests and attempts to apply rhyme detection to the phrases.
  + There is JavaScript that generally organizes the song lyrics into parts, with accompanying methods that can be applied to those parts.
* HTML, there is an index.html landing page.
  + Lays out the design of the page.
  + Contains some JavaScript that works with the drag and drop song elements.

Most of the work for the project seems to be related to writing the JavaScript. It seems as if the JavaScript comes into two parts, broadly: 1) logic, that models the song lyrics as if they were actual song elements. In other words I write JavaScript classes that act as phrases, bars, and so on and so forth. I have to model the code as if it were an actual song object. This makes it easier to work with. I suppose that’s what object-oriented programming is, though. Interestingly, this seems to bridge the line between metaphorical, and literal code. Instead of writing code that parses song lyrics and acts like it’s doing something, I have to write code that **is** song lyrics and actually does what it represents. In other words: representation = what it is. 2) code that parses lyrics and performs some regex stuff for ripping lyrics and stripping punctuation.

4.4 Schedule

A breakdown of the development schedule for this project.

4.4.1 GANTT Chart



4.4.2 Task/Resource Table

|  |  |  |
| --- | --- | --- |
| JavaScript | James | VIM/Notepad++/Sublime Text 3  Chromium/Google Chrome/Iceweasel |
| HTML | James | VIM/Notepad++/Sublime Text 3  Chromium/Google Chrome/Iceweasel |