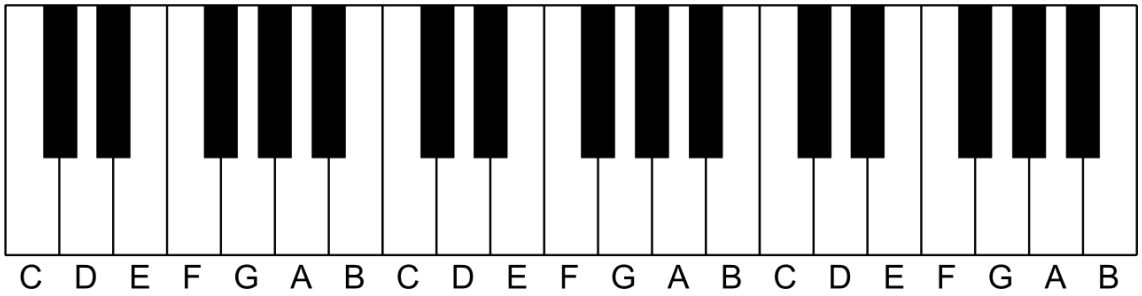
Noteable!

Project 2

Ana Beglova
Chris Wright
Nigel Stuart
Jonathan Kelley



Requirements Analysis

Stakeholders - source of requirements

- students (using the application)
- parents (presenting wants for the project)
- developers (in this case responsible for working out requirements)
- teachers (domain experts)

Functional Requirements

- The application will implement the following modes:
  - free play - to provide a keyboard the user can interact with to test sounds and ideas
  - quiz - to assess what the user has retained over the course of the tutorials
  - tutorial - to provide elementary note reading skills and the ability to apply them on a piano
  - song demos - to provide the user with a sense of how to take some of the notes they've learned and form a song

Requirements Analysis

Functional Requirements Cont…

- The application will display a keyboard that is able to sound notes with a mouse or keyboard

Nonfunctional Requirements

- The application is easily portable - moveable to different browsers
- Once the initial design is implemented, the application should be easy to maintain
- The application should be available (uptime) 99% of the time
- The application should rely on minimal external interfaces
    - internet connection
    - web browser - Firefox, Chrome, Opera 15, IE 9
    - computer mouse (keyboard has less accurate functionality)

Functional Requirements

Essential requirements

- quiz students on information learned
- provide instruction of note reading and where those notes fall on the piano

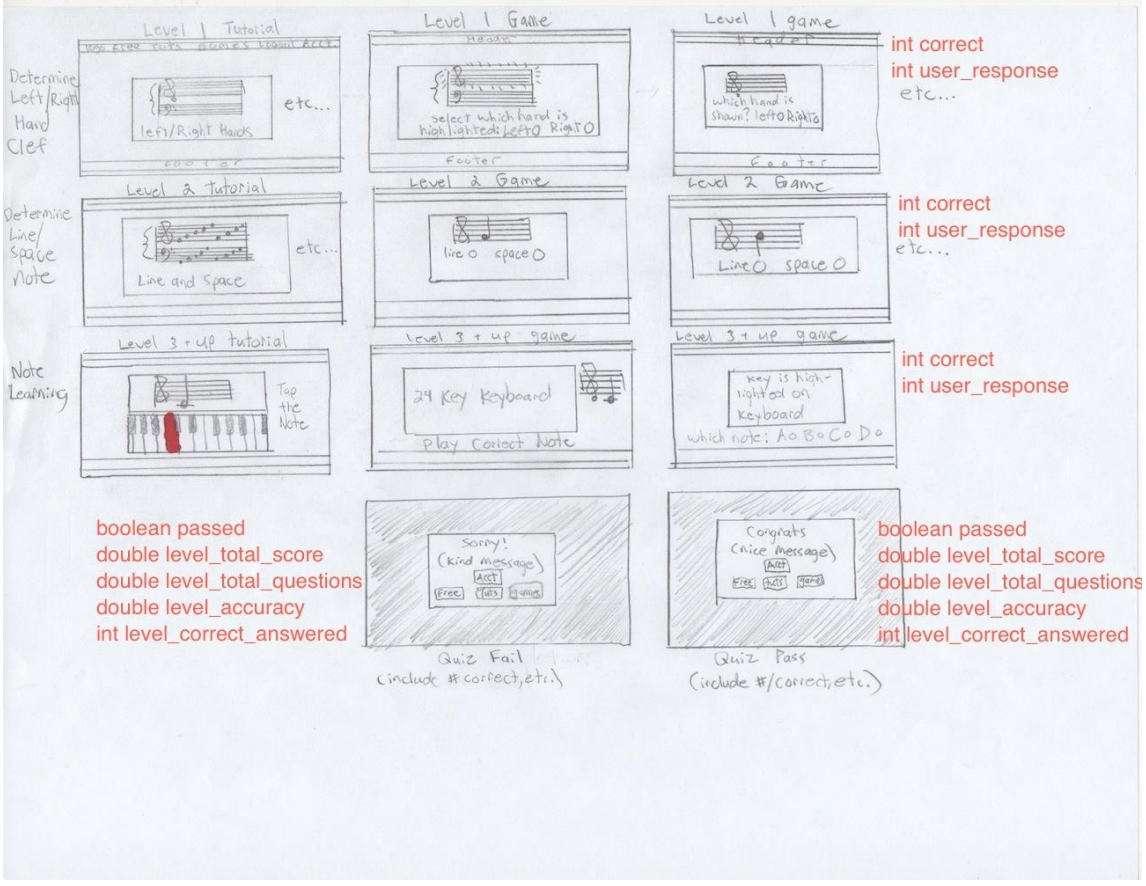
Desirable

- The application will be able to track progress through tutorial and quiz modes (save/load game)
- User login/statistics
- quiz scores/statistics

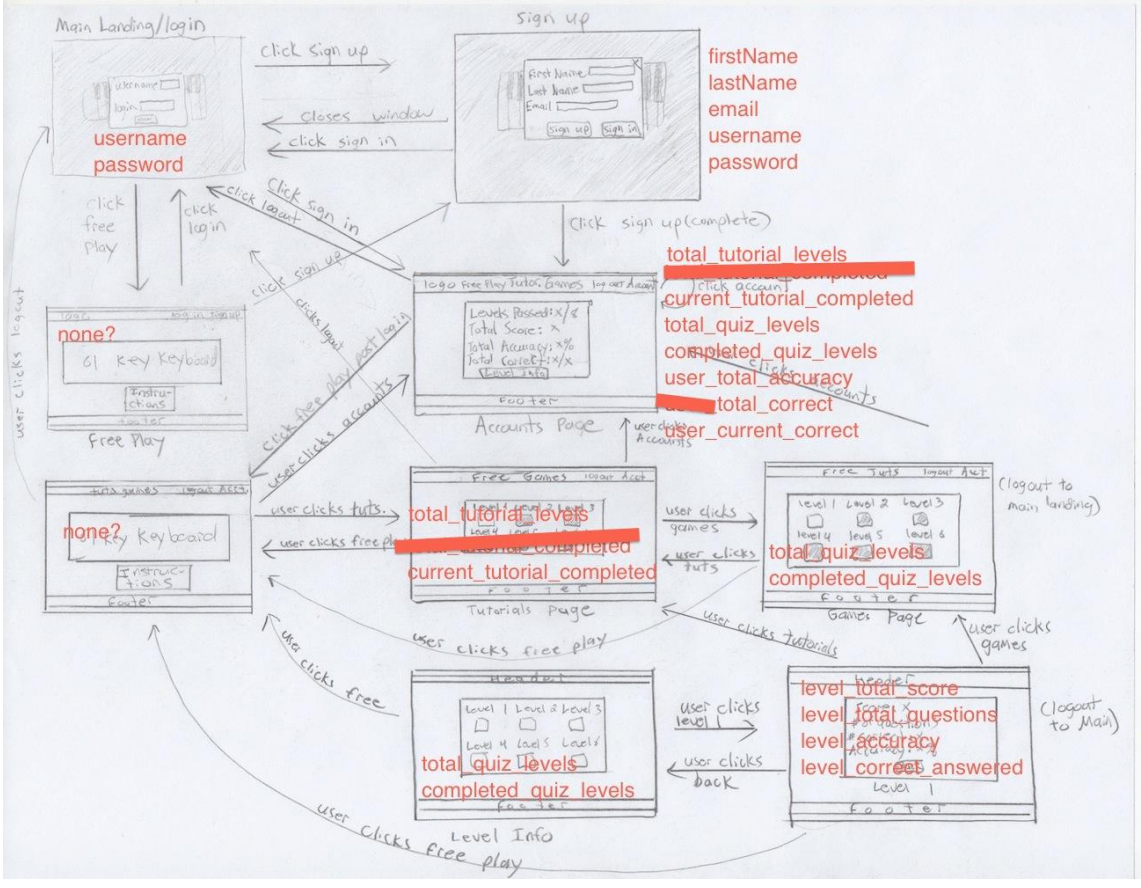
Optional requirements

- mobile
- rhythmic instruction
- the application will be able to record performance
- ability to change appearance
- change the sound of the instrument

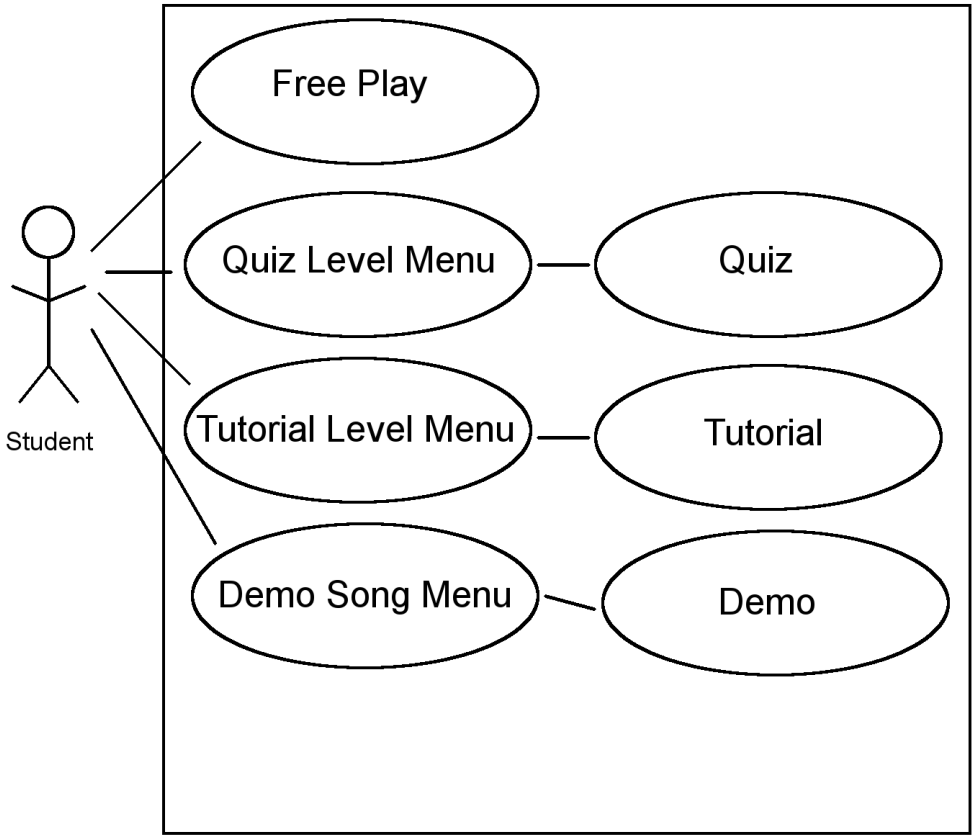
Requirements Analysis



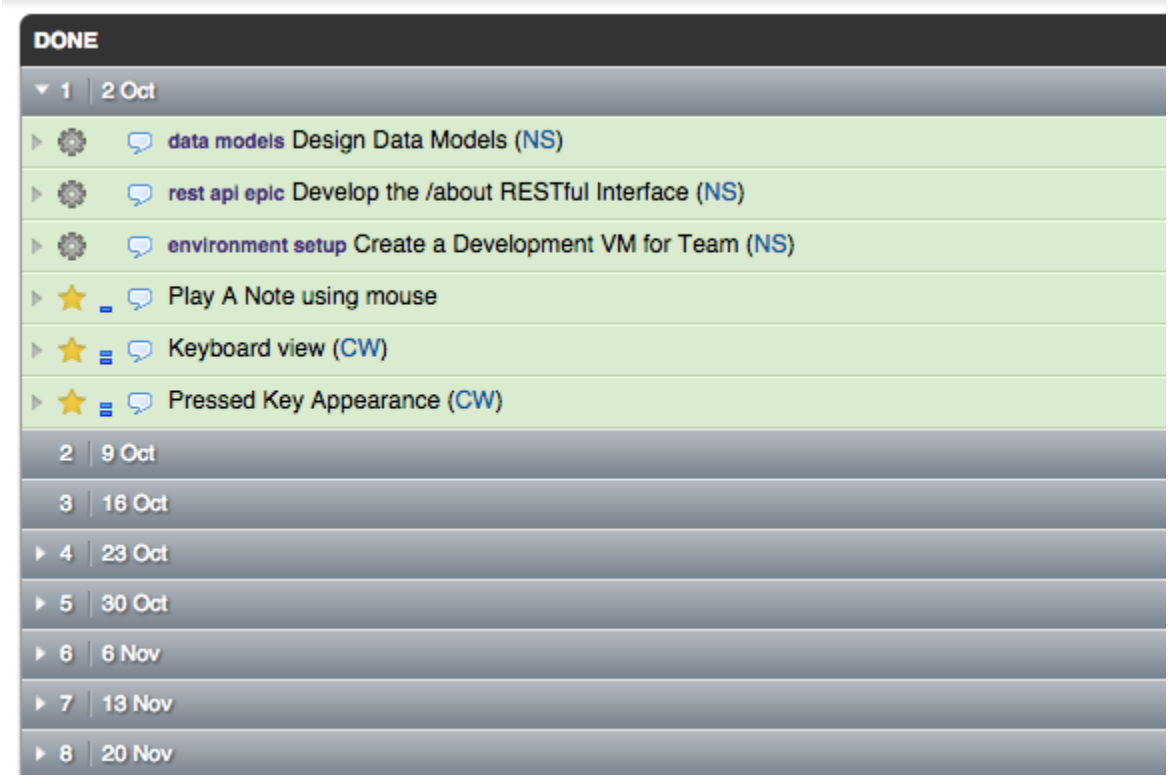
Requirements Analysis



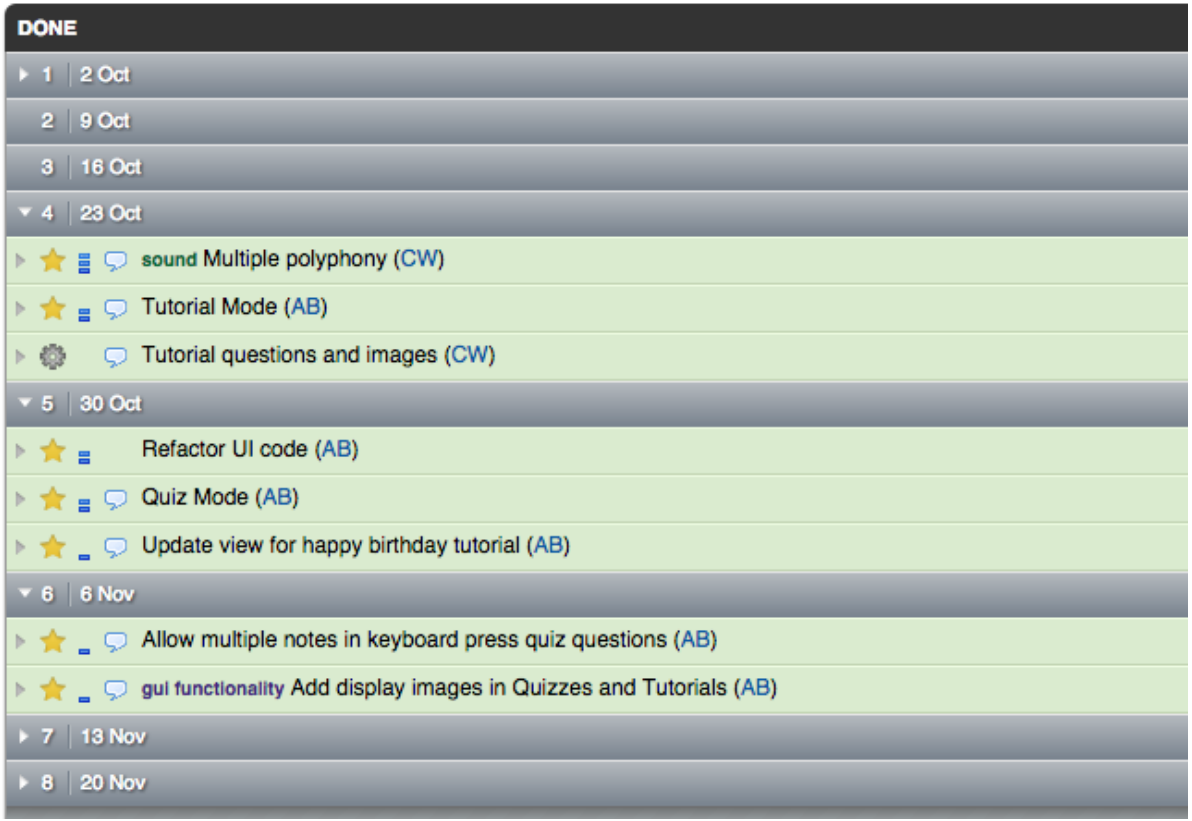
System Use Case Diagram



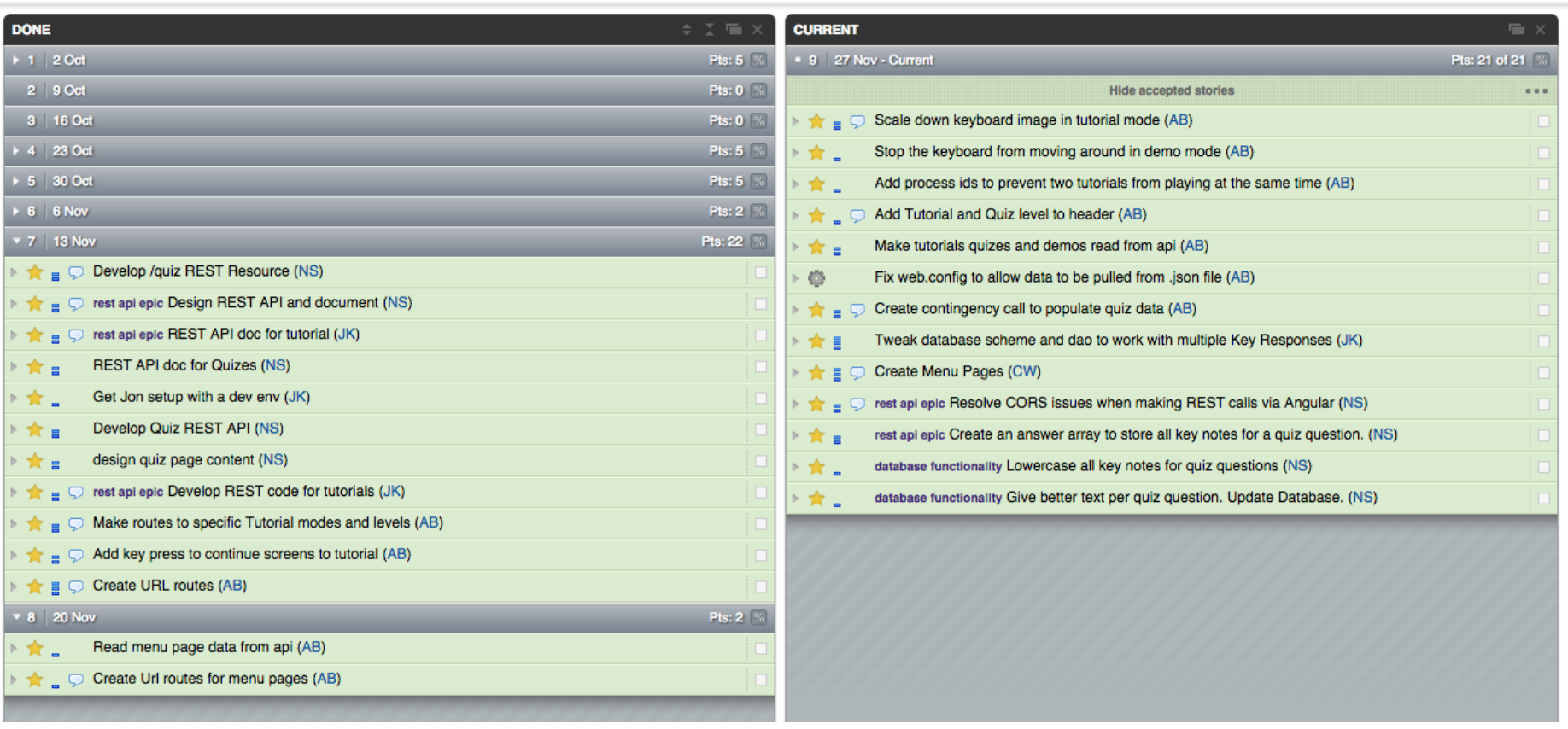
User Stories (1st Iteration)



User Stories (2nd Iteration)



User Stories (Final Iteration)



DONE

- 1 | 2 Oct — Pts: 5
- 2 | 9 Oct — Pts: 0
- 3 | 16 Oct — Pts: 0
- 4 | 23 Oct — Pts: 5
- 5 | 30 Oct — Pts: 5
- 6 | 6 Nov — Pts: 2
- 7 | 13 Nov — Pts: 22
  - Develop /quiz REST Resource (NS)
  - rest api epic Design REST API and document (NS)
  - rest api epic REST API doc for tutorial (JK)
  - REST API doc for Quizes (NS)
  - Get Jon setup with a dev env (JK)
  - Develop Quiz REST API (NS)
  - design quiz page content (NS)
  - rest api epic Develop REST code for tutorials (JK)
  - Make routes to specific Tutorial modes and levels (AB)
  - Add key press to continue screens to tutorial (AB)
  - Create URL routes (AB)
- 8 | 20 Nov — Pts: 2
  - Read menu page data from api (AB)
  - Create Url routes for menu pages (AB)

CURRENT

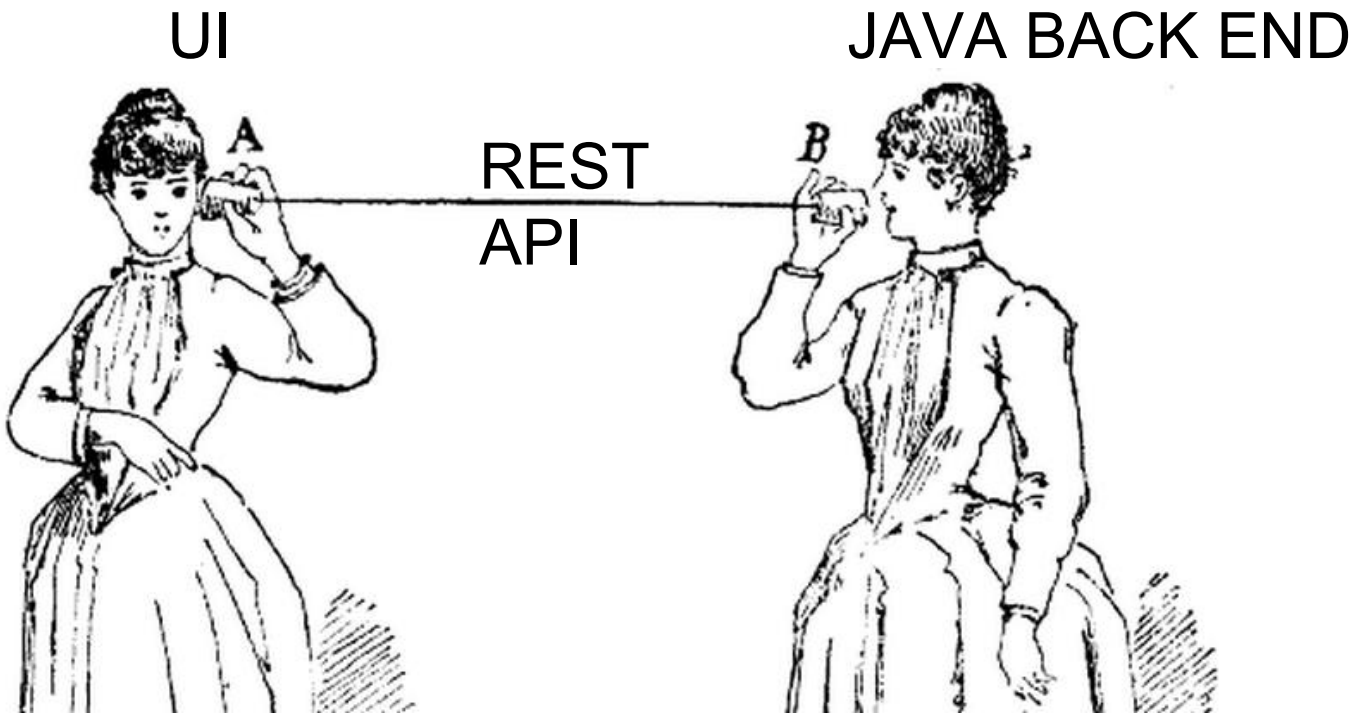
- 9 | 27 Nov - Current — Pts: 21 of 21

Hide accepted stories

- Scale down keyboard image in tutorial mode (AB)
- Stop the keyboard from moving around in demo mode (AB)
- Add process ids to prevent two tutorials from playing at the same time (AB)
- Add Tutorial and Quiz level to header (AB)
- Make tutorials quizes and demos read from api (AB)
- Fix web.config to allow data to be pulled from .json file (AB)
- Create contingency call to populate quiz data (AB)
- Tweak database scheme and dao to work with multiple Key Responses (JK)
- Create Menu Pages (CW)
- rest api epic Resolve CORS issues when making REST calls via Angular (NS)
- rest api epic Create an answer array to store all key notes for a quiz question. (NS)
- database functionality Lowercase all key notes for quiz questions (NS)
- database functionality Give better text per quiz question. Update Database. (NS)

Tools and Techniques

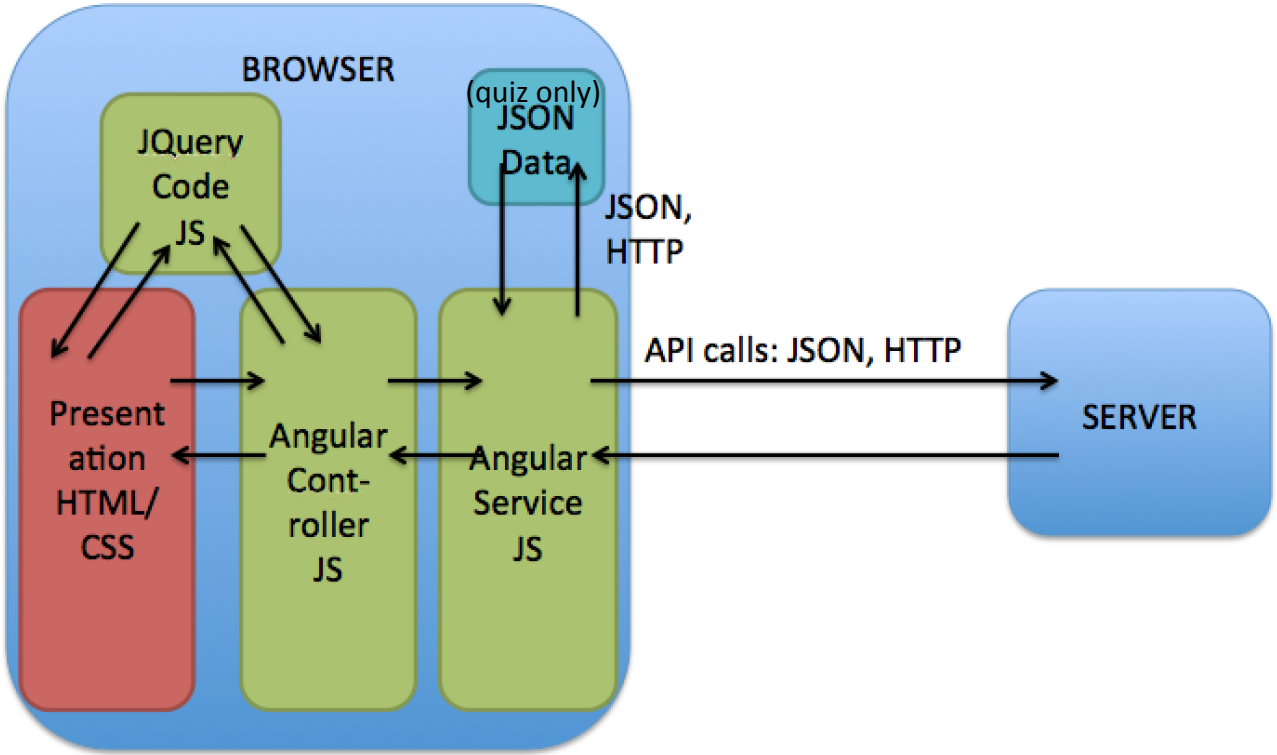
Architecture Overview



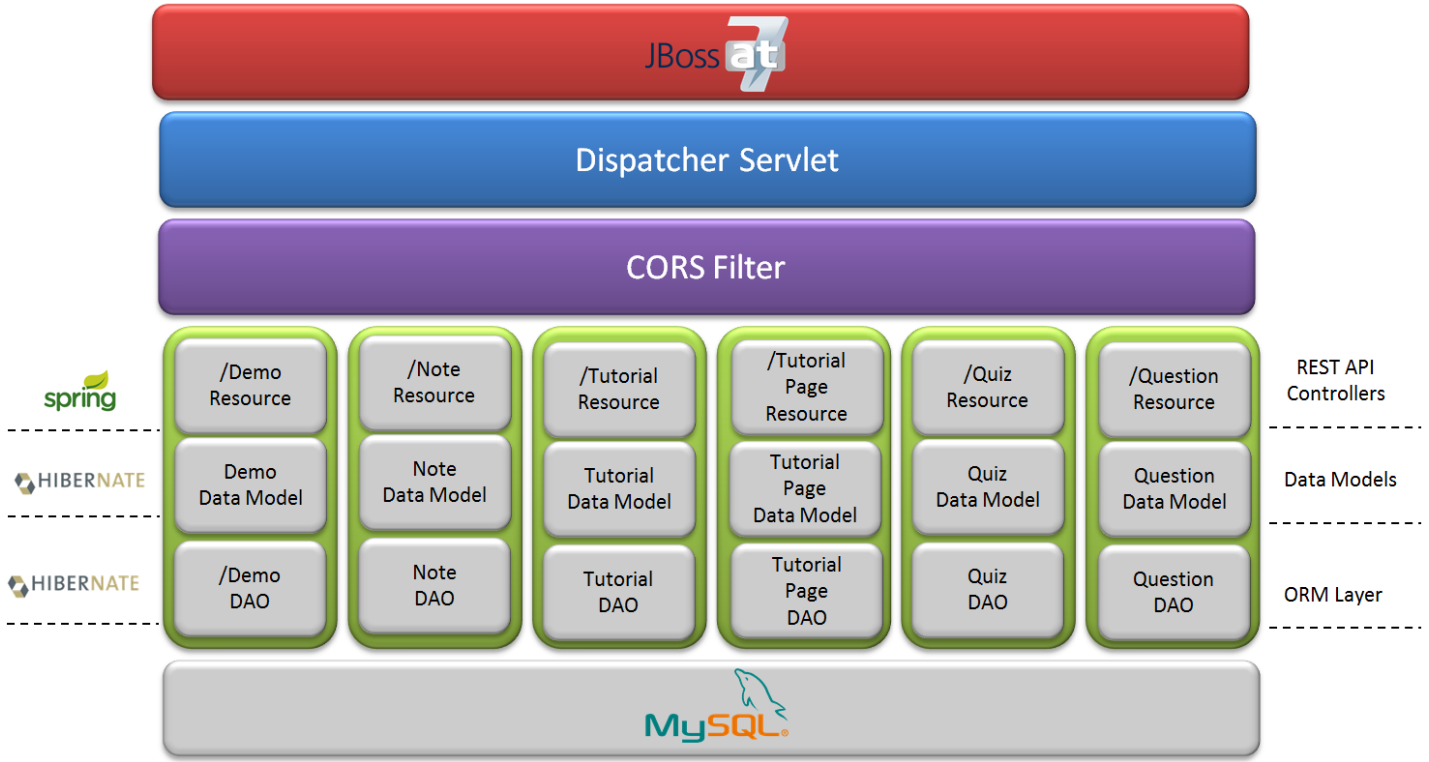
UI Architecture

- Two page app - but with lots of javascript to change content!
  - level_selection.html page for all the selection menus
  - piano.html for all keyboard functionality
- Data comes from REST API
- Angular controllers for both pages
  - Custom routes for navigation
  - Two way data binding
- Process Hashes

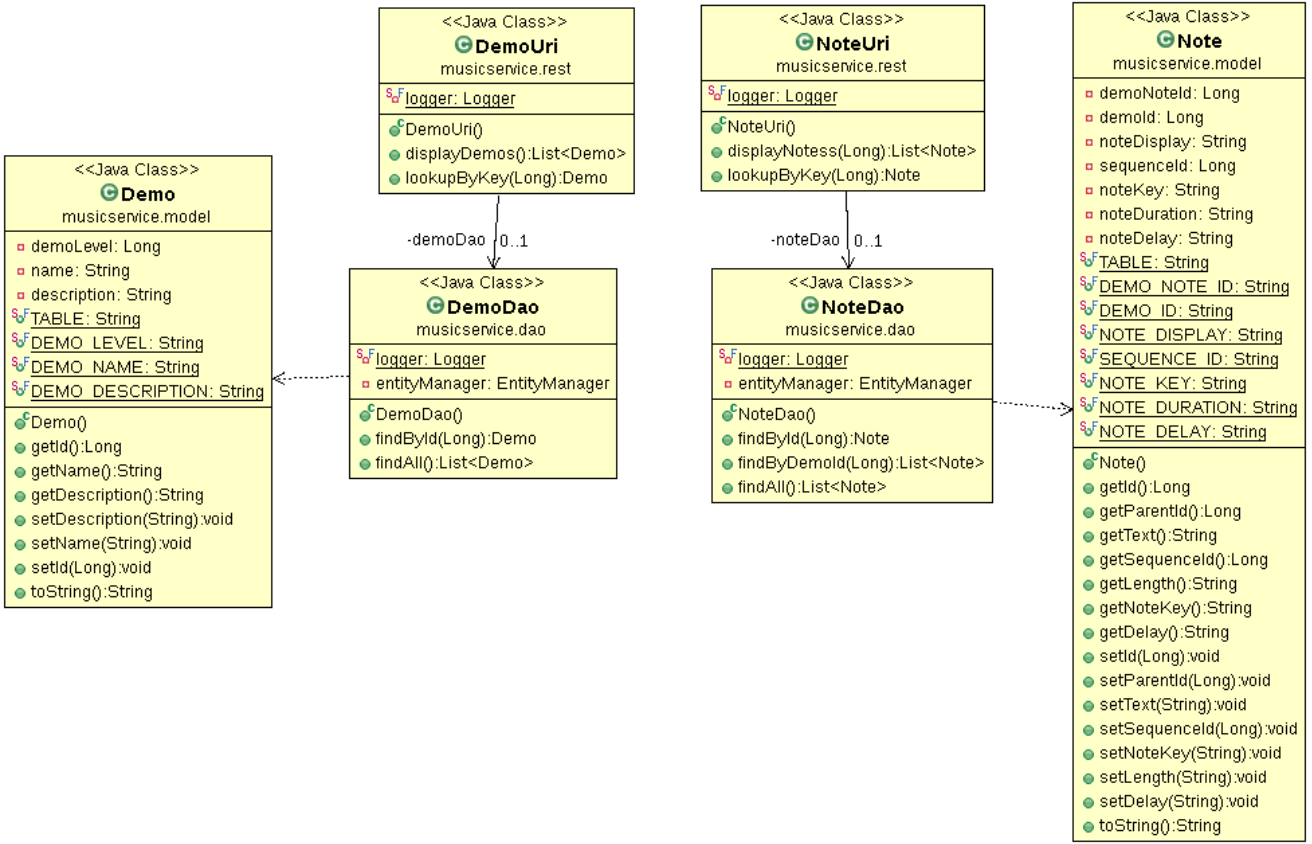
UI Architecture



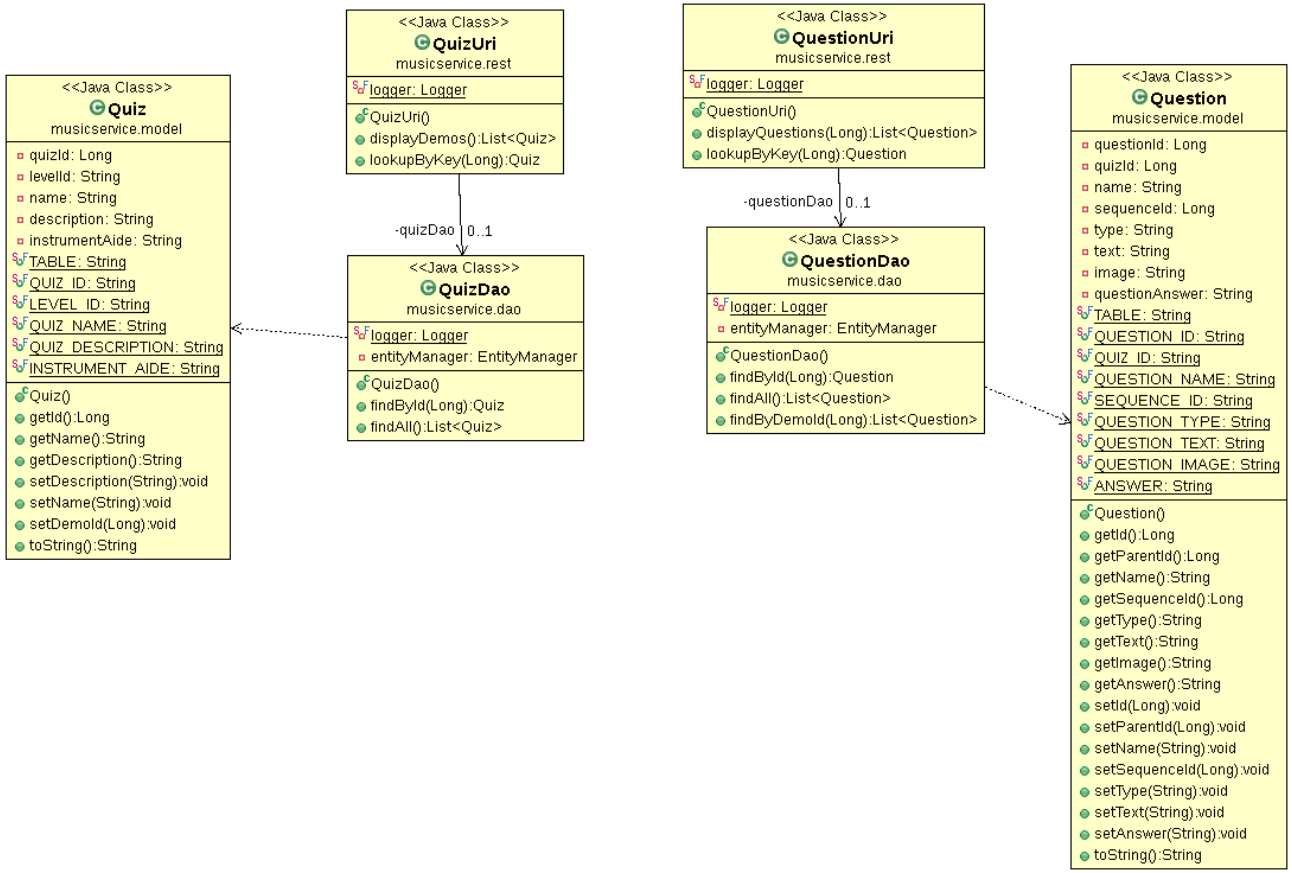
Backend Architecture



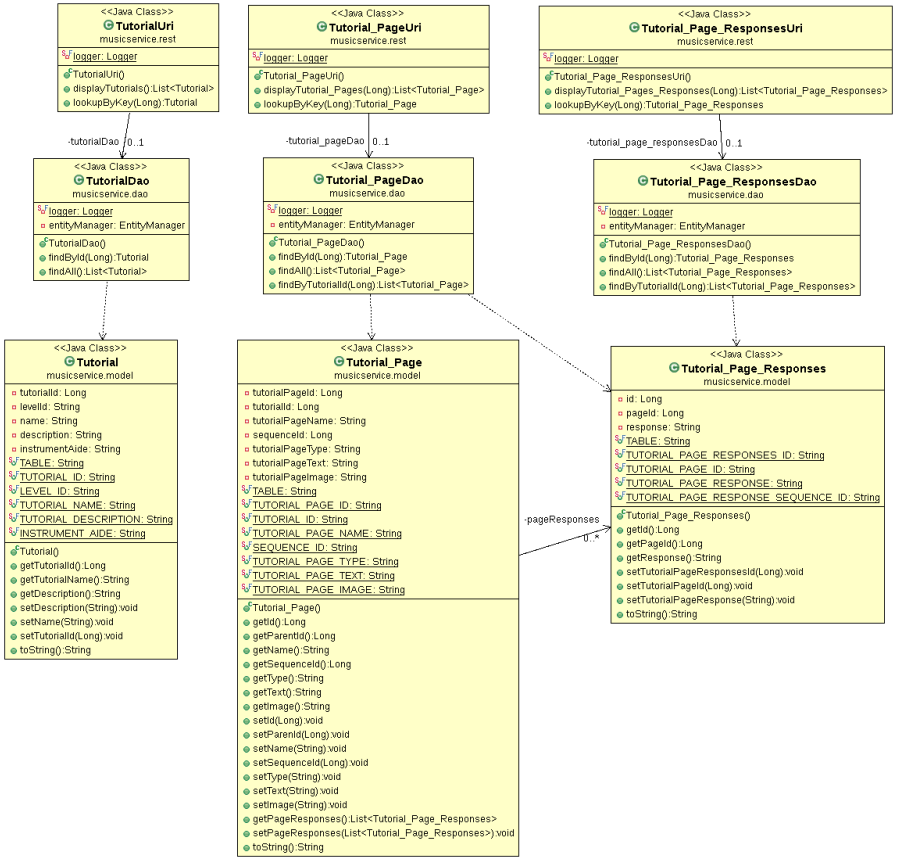
Class Diagram - Demo



Class Diagram - Quiz



Class Diagram - Tutorial



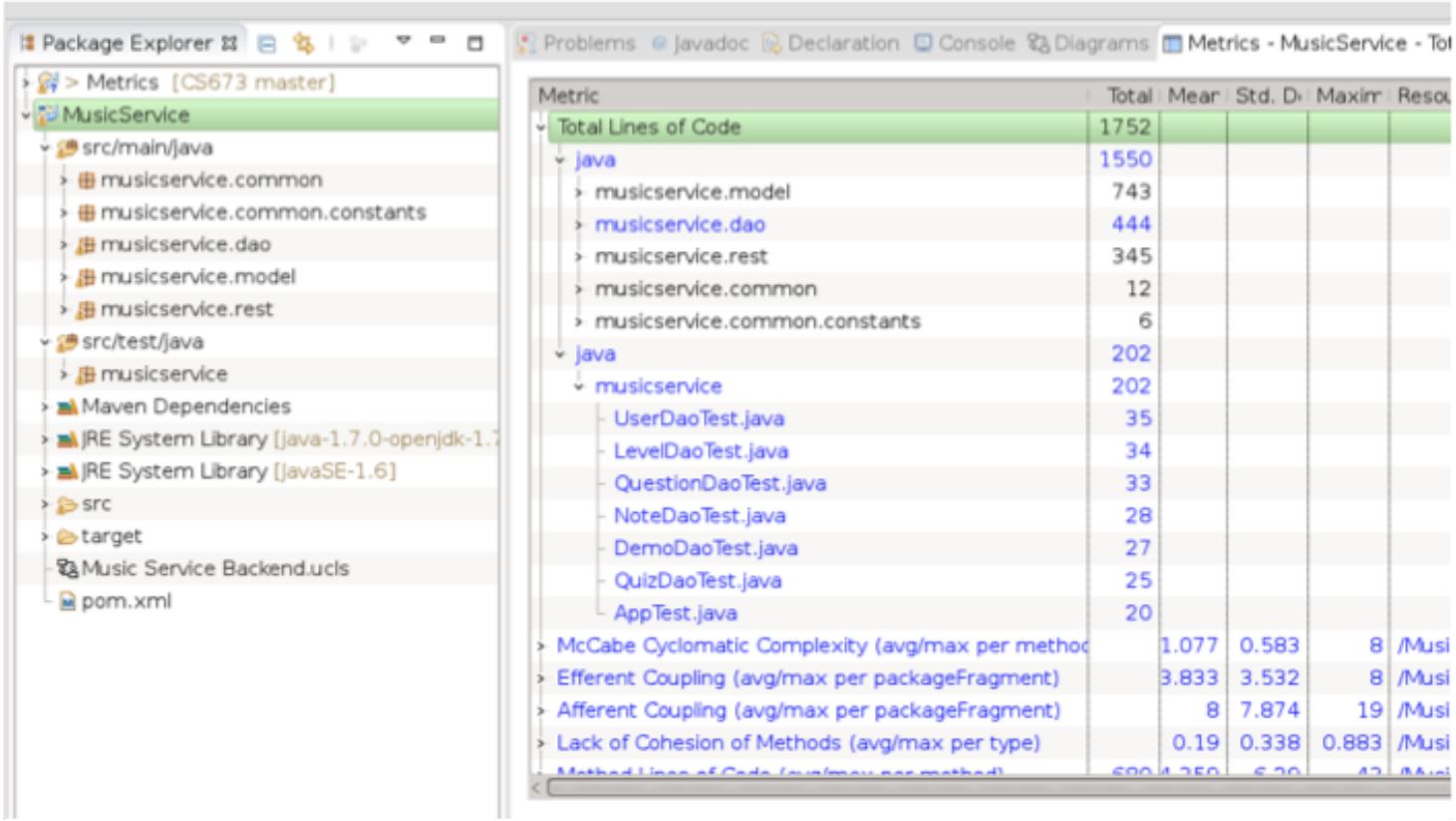
REST API

http://keyboard.cloudapp.net:3010/MusicService/user
http://keyboard.cloudapp.net:3010/MusicService/quiz
http://keyboard.cloudapp.net:3010/MusicService/demo
http://keyboard.cloudapp.net:3010/MusicService/question
http://keyboard.cloudapp.net:3010/MusicService/note
http://keyboard.cloudapp.net:3010/MusicService/tutorial
http://keyboard.cloudapp.net:3010/MusicService/tutorial_page
http://keyboard.cloudapp.net:3010/MusicService/tutorial_page?tutorialId=1
http://keyboard.cloudapp.net:3010/MusicService/question?quizId=1
http://keyboard.cloudapp.net:3010/MusicService/note?demoId=1

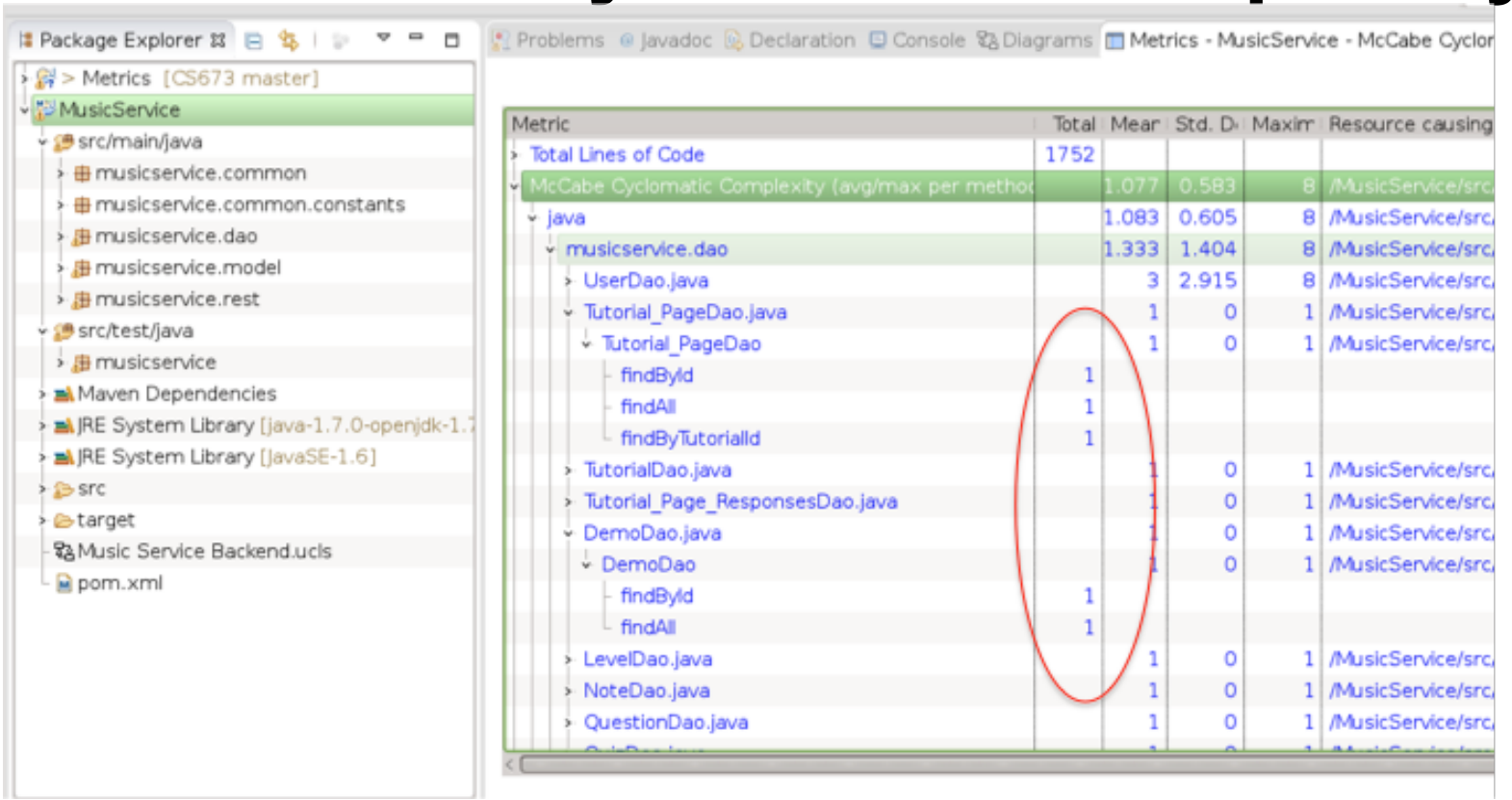
Quality Metrics

- Lines of Code
- Cyclomatic Complexity
- Productivity
- Deficiency Tracking

Backend Lines of Code



Back-end Cyclomatic Complexity



Front-end Lines of Code

Line	Function	Statements	Lines	Comment Lines	Comment%	Branches	Depth	Cyclomatic Complexity
1	[[code]]	784	926	16	1.73%	0	0	1
1	(Anonymous1)	5	3	0	0%	0	0	1
2	(Anonymous1).(Anonymous1)	1	2	0	0%	0	0	1
7	updateLevelsPage	23	28	0	0%	5	3	6
37	(Anonymous2)	38	27	0	0%	0	0	1
42	(Anonymous2).(Anonymous1)	6	8	0	0%	3	1	4
53	(Anonymous2).setNavigationData	23	11	0	0%	3	1	4
56	(Anonymous2).setNavigationData.(Anonymous1)	2	1	0	0%	0	0	1
59	(Anonymous2).setNavigationData.(Anonymous2)	2	1	0	0%	0	0	1
62	(Anonymous2).setNavigationData.(Anonymous3)	2	1	0	0%	0	0	1
71	(Anonymous3)	7	11	0	0%	0	0	1
86	(Anonymous4)	16	19	0	0%	0	0	1
87	(Anonymous4).tutorial_overview	2	2	0	0%	0	0	1
91	(Anonymous4).quiz_overview	2	2	0	0%	0	0	1
95	(Anonymous4).demo_overview	2	2	0	0%	0	0	1
101	(Anonymous4).(Anonymous1)	1	2	0	0%	0	0	1
103	(Anonymous4).(Anonymous2)	0	2	0	0%	0	0	1
109	(Anonymous5)	153	89	2	2.25%	0	1	1
112	(Anonymous5).(Anonymous1)	3	4	0	0%	0	0	1
117	(Anonymous5).(Anonymous2)	3	4	0	0%	0	0	1
122	(Anonymous5).(Anonymous3)	1	2	0	0%	0	0	1
194	(Anonymous5).(Anonymous4)	12	3	0	0%	0	1	1

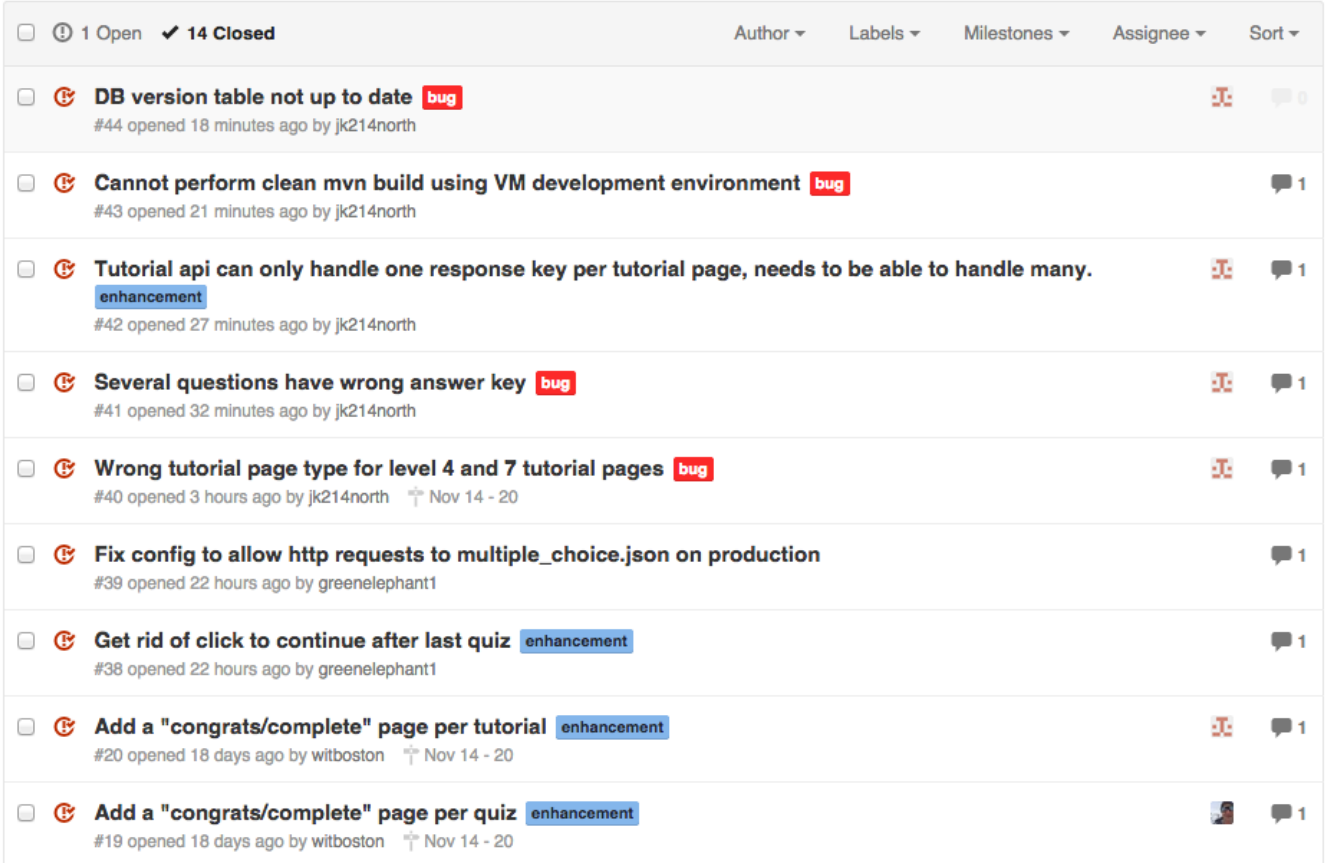
Front-end Cyclomatic Complexity

Line	Function	Statements	Lines	Comment Lines	Comment%	Branches	Depth	Cyclomatic Complexity
206	stopNote	2	4	0	0%	0	0	1
213	key_down	1	2	0	0%	0	0	1
217	(Anonymous6)	229	284	5	1.76%	0	0	1
239	(Anonymous6).setMode	32	38	1	2.63%	8	4	9
263	(Anonymous6).setMode.(Anonymous1)	1	3	0	0%	0	2	1
276	(Anonymous6).setMode.(Anonymous2)	1	1	0	0%	0	4	1
282	(Anonymous6).setLevel	5	6	1	16.67%	1	1	2
290	(Anonymous6).setMCChoice	1	2	0	0%	0	0	1
294	(Anonymous6).recieveKeyboardPress	28	28	0	0%	8	3	9
327	(Anonymous6).recieveClickToContinue	14	18	0	0%	6	4	7
350	(Anonymous6).(Anonymous1)	6	8	0	0%	4	1	5
361	(Anonymous6).checkKeyboardPressAnswer	6	10	0	0%	1	1	3
373	(Anonymous6).wrongAnswerDisplay	4	3	0	0%	0	0	1
378	(Anonymous6).correctAnswerDisplay	4	3	0	0%	0	0	1
383	(Anonymous6).iterateTutorial	20	19	0	0%	4	2	5
406	(Anonymous6).setDisplayText	7	6	0	0%	2	2	3
416	(Anonymous6).setDisplayImage	7	6	0	0%	2	2	3
426	(Anonymous6).simulateKeyPress	4	5	0	0%	1	1	2
433	(Anonymous6).playDemo	36	40	0	0%	0	0	1
439	(Anonymous6).playDemo.playNextNote	31	30	0	0%	2	2	3
475	(Anonymous6).iterateQuiz	22	19	0	0%	2	2	3

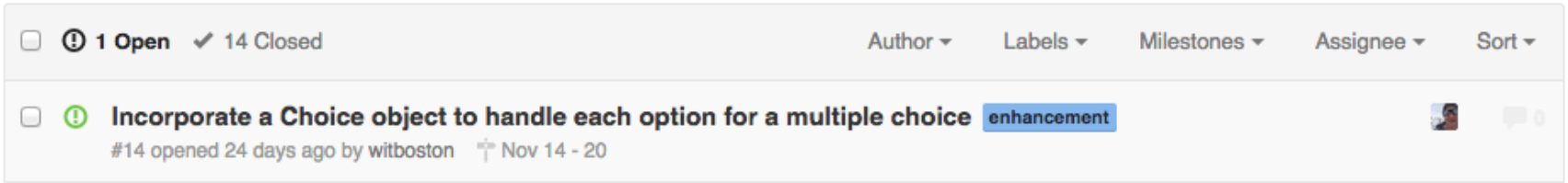
Productivity

- Front End:
  - LOC = 926
  - LOC/Hour = 4.75
  - Does not include Angular or JQuery java Script
- Back End:
  - LOC = 1752
  - LOC/Hour = 6.92

Deficiency Tracking



Deficiency Tracking



Multifaceted Test Strategy

- Unit Testing
  - Manual
  - Individual Developers
  - Front-end testing using temporary data structures
  - Browser based back-end testing
  - Requirements testing
- Maven Build Testing
  - Automated
  - Limited Regression Testing
  - Back-end only
- End to End Testing
  - Manual
  - Production environment
  - Complete free play, tutorial, quiz and demo level coverage
  - Automated/Selenium - Work In Progress

Maven Build Test Example

```
@Test
public void testGetAllTutoriaPages()
{
      List<Tutorial_Page> pages = tutorial_PageDao.findAll();
      assertEquals(37, pages.size());
      assertEquals("Conclusion", pages.get(37).getName());
}
    // Get a the specific tutorial page from the test database.
@Test
public void testGetTutoriaPage()
{
      long id = 37;
      Tutorial_Page page = tutorial_PageDao.findById(id);
      assertEquals(id, page.getId());
      assertEquals("Conclusion", page.getName());
}
```

```
[INFO] ------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------
[INFO] Total time: 8.920s
[INFO] Finished at: Tue Nov 11 10:10:00 CET 2014
[INFO] Final Memory: 14M/120M
[INFO] ------------------------------------------
```

Maven Test Example Continued

```
//Get a the specific tutorial page from the test database.
    @Test
    public void testGetTutoriaPagesByTutorialId()
    {
        List<Tutorial_Page> pages = tutorial_PageDao.findByTutorialId((long) 8);
        assertEquals(3, pages.size());
        assertEquals("Conclusion", pages.get(3).getName());

     }
```

Release Process - Database

- Audit table to track database versions.
  - Needed to ensure changes are not lost.
  - Helpful when debugging.
  - Create and alter scripts for database upgrades

- schema_00100_alter.sql
- schema_00100_alterdata.sql
- schema_00100_create.sql
- schema_00200_alter.sql
- schema_00200_alterdata.sql
- schema_00200_create.sql
- schema_00300_alter.sql
- schema_00300_alterdata.sql
- schema_00300_create.sql
- schema_00400_create.sql
- schema_00500_create.sql

```
VALUES (1, '1.0', 'Added user table and db_version table.');
VALUES (2, '2.0', 'Added core tables for entire projet.');
VALUES (3, '3.0', 'Added tutorial, tutorial_pages, and tutorial_page_responses table.');
VALUES (4, '4.0', 'Modified note duration to be a varchar. Lower cased music notes. Tweaked one tu
VALUES (5, '5.0', 'Tweaked some tutorial page  types to press_key and some question answers.');
```

Release Process - Production

- Used git “development” branch to merge our code.
- Pushed to production when code was ready for release.
- Production was live on keyboard.cloudapp.net
    - Allowed GUI team and Backend team to focus on their areas

- Use of Hibernate (Object Relational Mapping) allows for database interchangeability.
- Data Models passed and returned per method allows for greater flexibility.
- Unit tests ensure Java build is valid as new features are added.

Challenges

- Personnel issues
- Learning new tools and technologies
- Communication and coordination
- Unforeseen technical challenges

Failures and Success

We made an attractive and useful app!*

*But we had to drop some functionality we were excited about

Lessons Learned

- Division of labor is efficient but risky
- Always have a back up plan
- Being disciplined with refactoring and process helps avoid bugs later
- Software development takes more time than expected
- Use of data models from the very start make development easier down the road.
- Angular is awesome!

DEMO

THE END