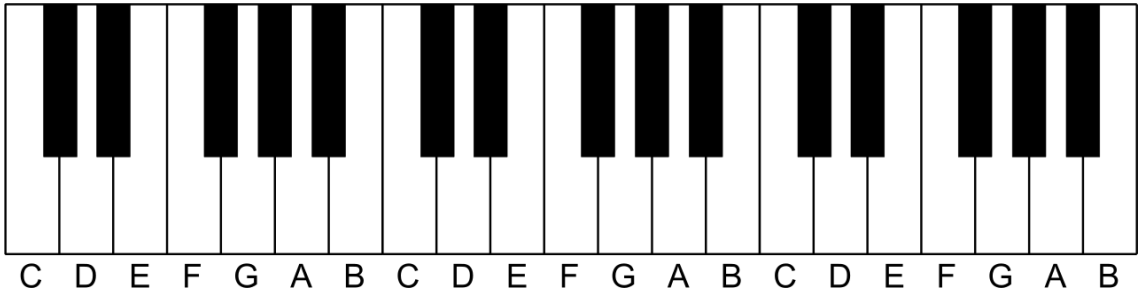
Noteable!

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

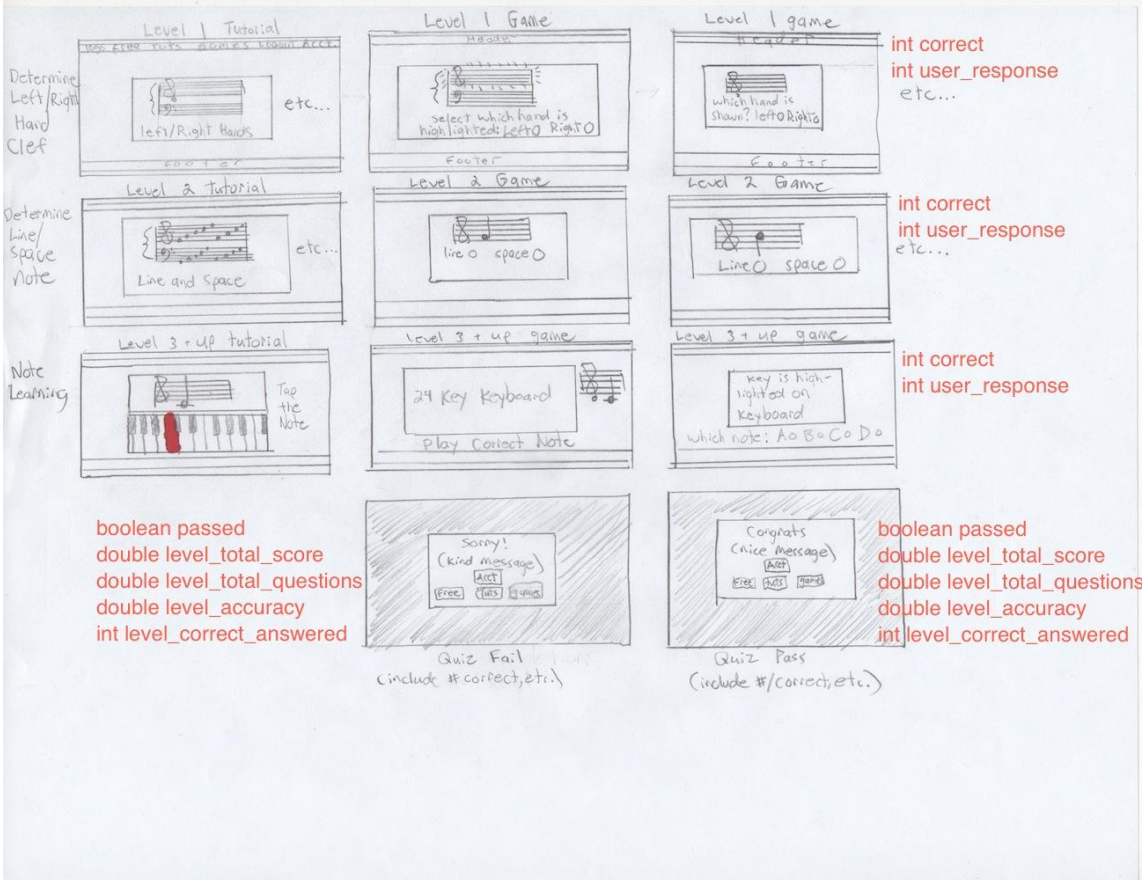
Desirable

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

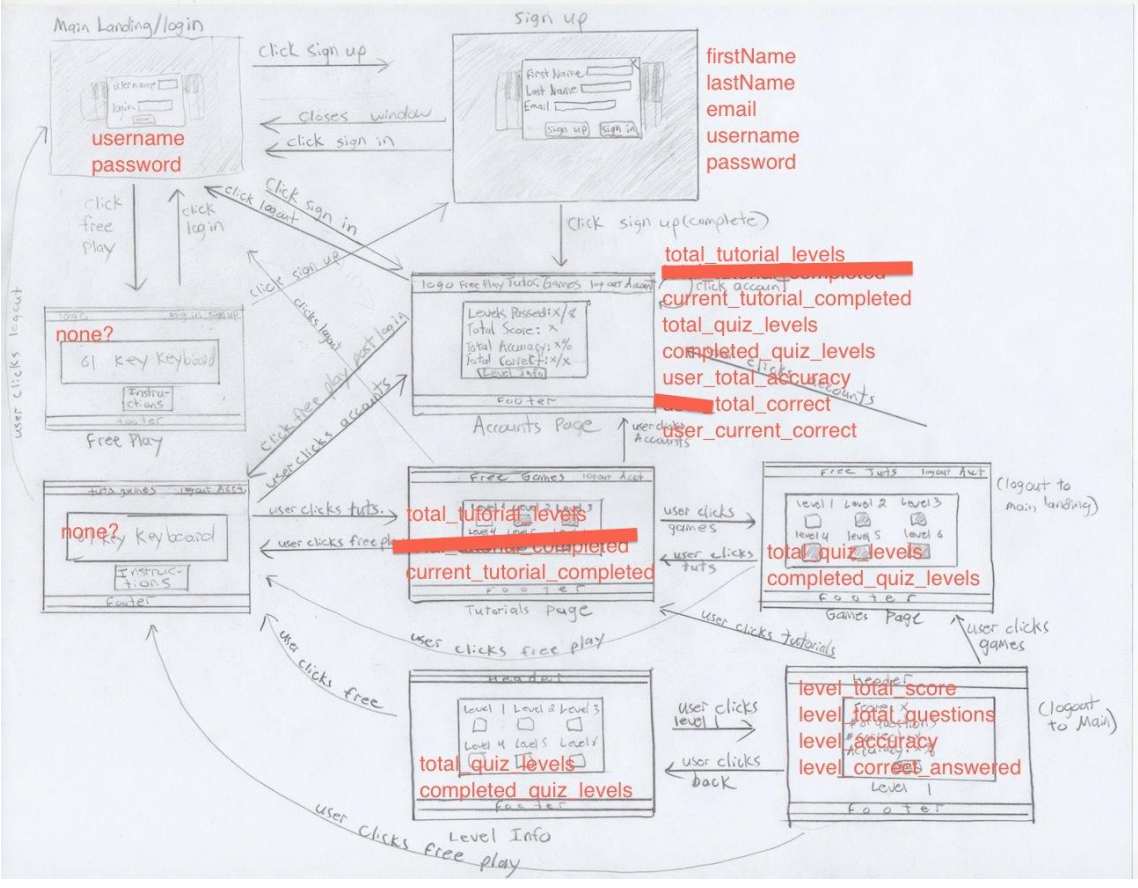
Optional requirements

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

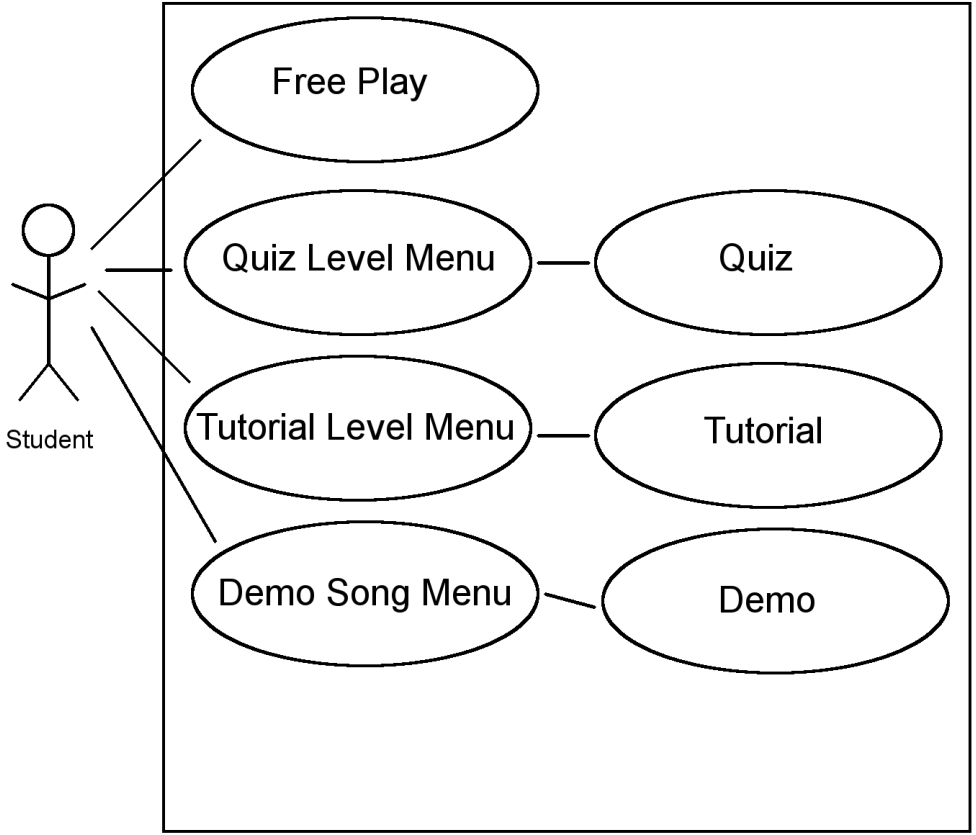
Requirements Analysis



Requirements Analysis



System Use Case Diagram



User Stories (1st Iteration)

DONE

1	2 Oct	
		data models Design Data Models (NS)
		rest api epic Develop the /about RESTful Interface (NS)
		environment setup Create a Development VM for Team (NS)
		Play A Note using mouse
		Keyboard view (CW)
		Pressed Key Appearance (CW)
2	9 Oct	
3	16 Oct	
4	23 Oct	
5	30 Oct	
6	6 Nov	
7	13 Nov	
8	20 Nov	

User Stories (2nd Iteration)

DONE

- 1 | 2 Oct
- 2 | 9 Oct
- 3 | 16 Oct
- 4 | 23 Oct
  - sound Multiple polyphony (CW)
  - Tutorial Mode (AB)
  - Tutorial questions and images (CW)
- 5 | 30 Oct
  - Refactor UI code (AB)
  - Quiz Mode (AB)
  - Update view for happy birthday tutorial (AB)
- 6 | 6 Nov
  - Allow multiple notes in keyboard press quiz questions (AB)
  - gui functionality Add display images in Quizzes and Tutorials (AB)
- 7 | 13 Nov
- 8 | 20 Nov

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

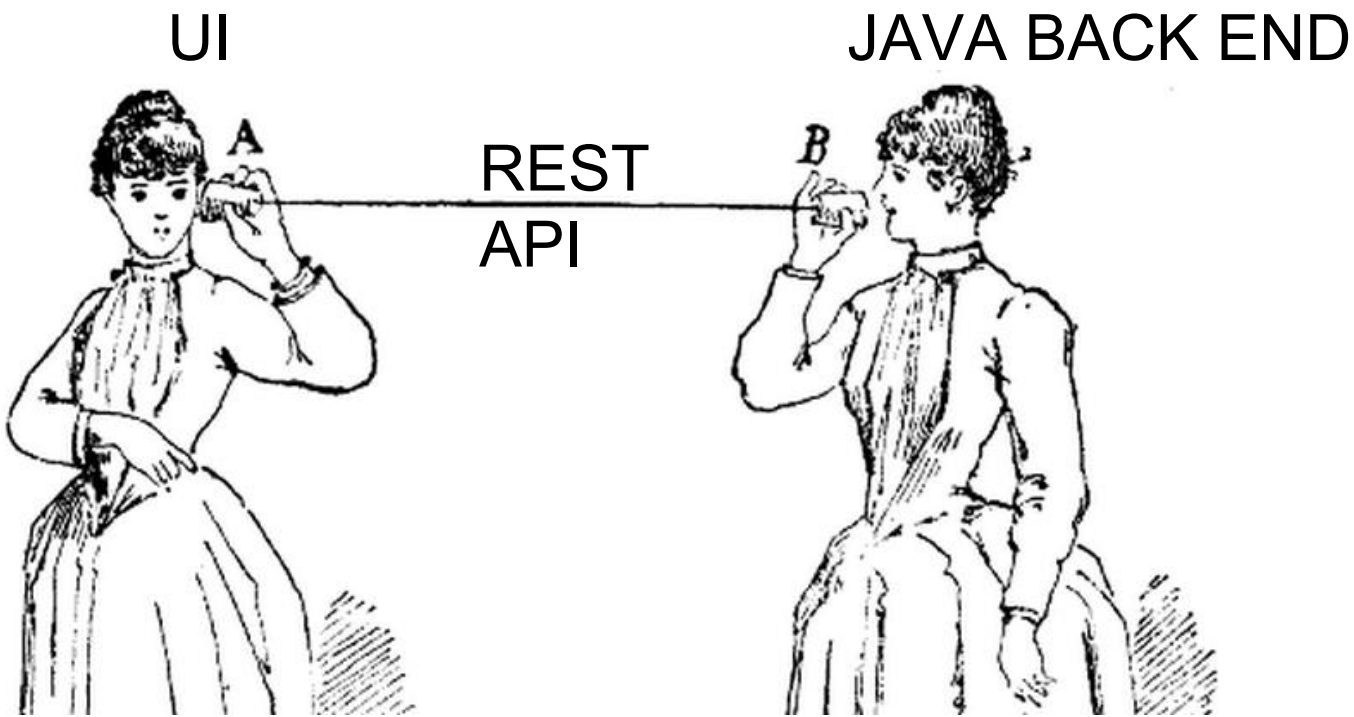
9	27 Nov - Current	Pts: 21 of 21

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

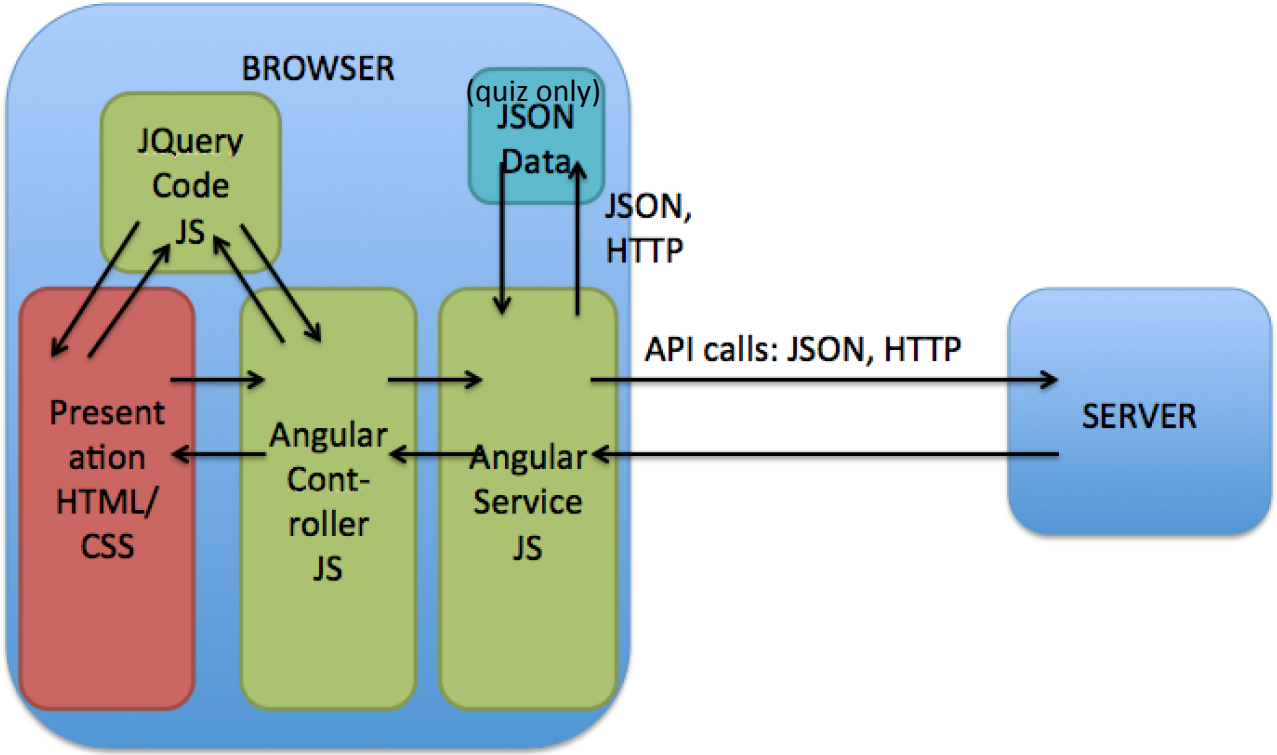
Architecture Overview



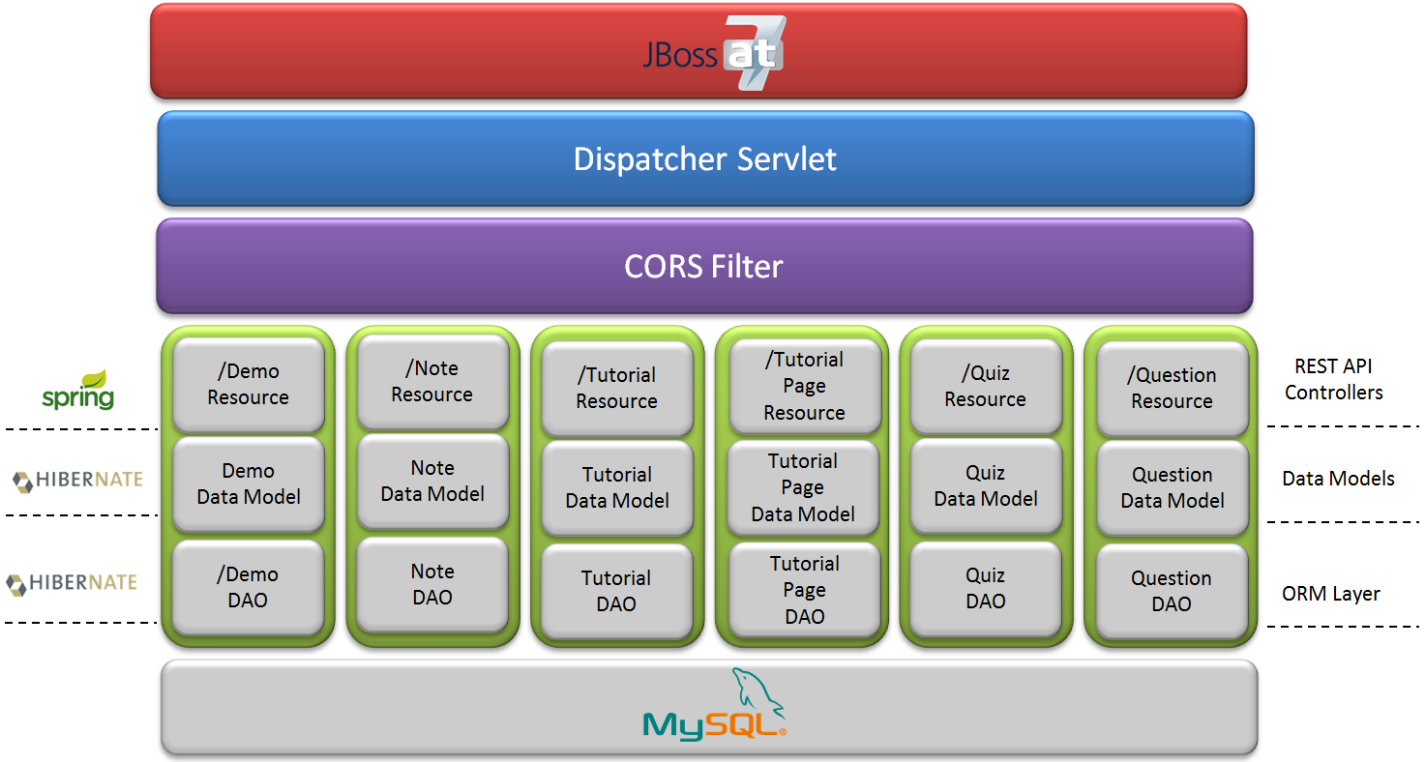
UI Architecture

- 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

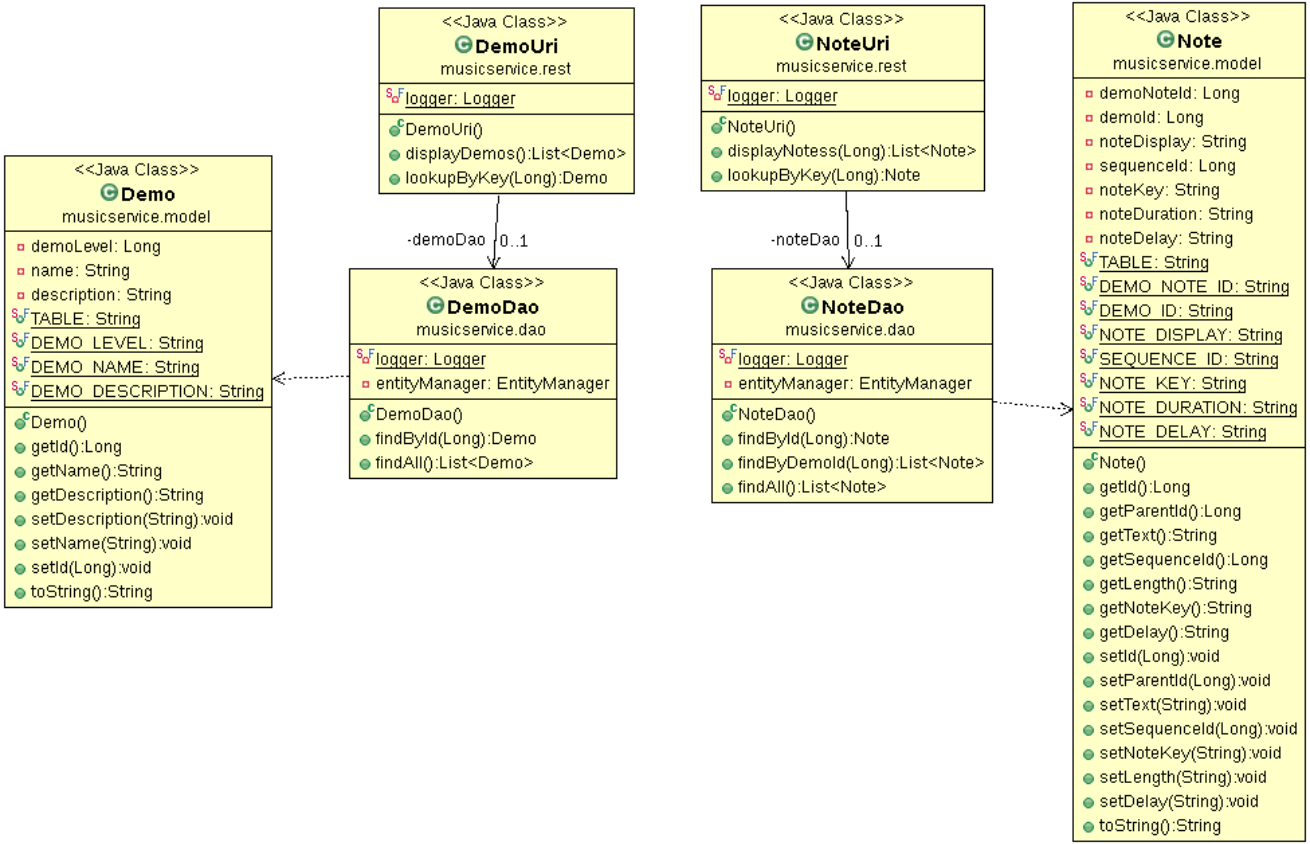
UI Architecture



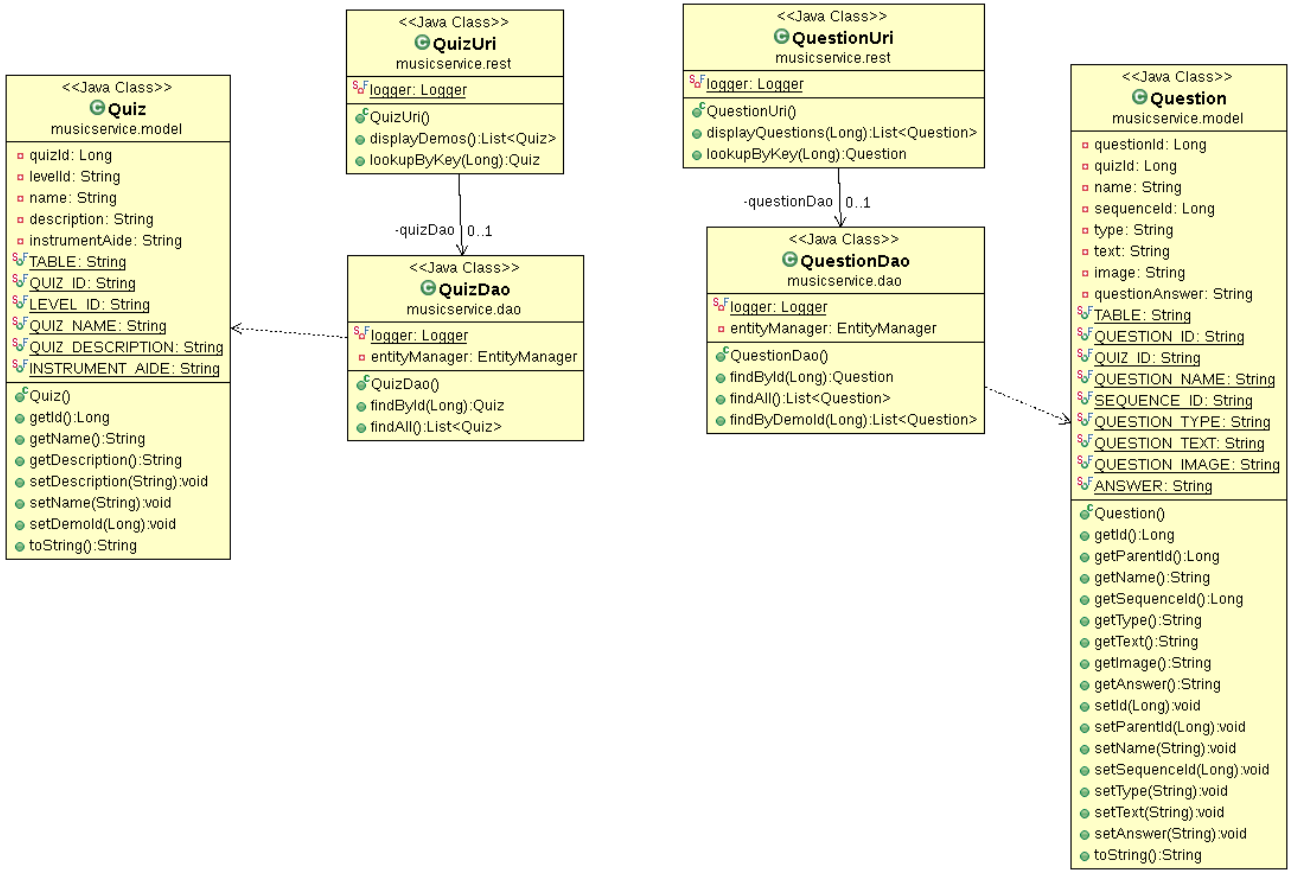
Backend Architecture



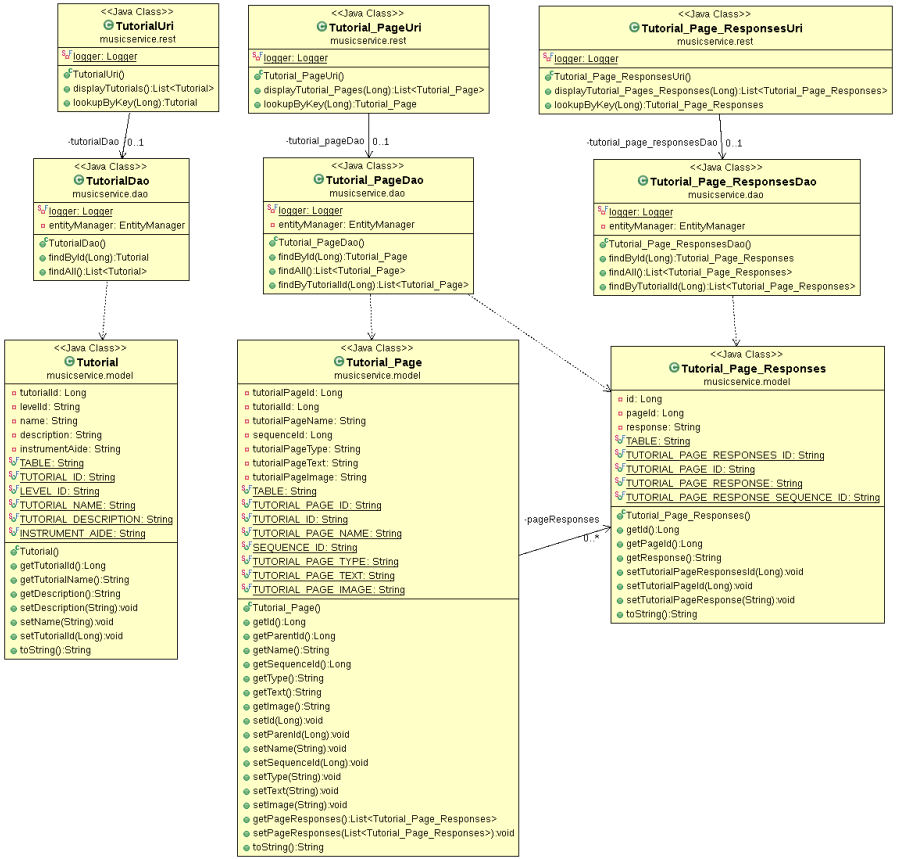
Class Diagram - Demo



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

Package Explorer

- > Metrics [CS673 master]
- MusicService
  - src/main/java
    - musicservice.common
    - musicservice.common.constants
    - musicservice.dao
    - musicservice.model
    - musicservice.rest
  - src/test/java
    - musicservice
  - Maven Dependencies
  - JRE System Library [java-1.7.0-openjdk-1.7
  - JRE System Library [JavaSE-1.6]
  - src
  - target
  - Music Service Backend.ucls
  - pom.xml

Problems | Javadoc | Declaration | Console | Diagrams | Metrics - MusicService - Tot

Metric	Total	Mean	Std. D	Maxim	Resou
Total Lines of Code	1752				
java	1550				
musicservice.model	743				
musicservice.dao	444				
musicservice.rest	345				
musicservice.common	12				
musicservice.common.constants	6				
java	202				
musicservice	202				
UserDaoTest.java	35				
LevelDaoTest.java	34				
QuestionDaoTest.java	33				
NoteDaoTest.java	28				
DemoDaoTest.java	27				
QuizDaoTest.java	25				
AppTest.java	20				
McCabe Cyclomatic Complexity (avg/max per method		1.077	0.583	8	/Musi
Efferent Coupling (avg/max per packageFragment)		3.833	3.532	8	/Musi
Afferent Coupling (avg/max per packageFragment)		8	7.874	19	/Musi
Lack of Cohesion of Methods (avg/max per type)		0.19	0.338	0.883	/Musi

Back-end Cyclomatic Complexity

Package Explorer

- > Metrics [CS673 master]
- MusicService
  - src/main/java
    - musicservice.common
    - musicservice.common.constants
    - musicservice.dao
    - musicservice.model
    - musicservice.rest
  - src/test/java
    - musicservice
  - Maven Dependencies
  - JRE System Library [java-1.7.0-openjdk-1.7
  - JRE System Library [JavaSE-1.6]
  - src
  - target
  - Music Service Backend.ucls
  - pom.xml

Problems | Javadoc | Declaration | Console | Diagrams | Metrics - MusicService - McCabe Cyclor

Metric	Total	Mear	Std. D	Maxim	Resource causing
Total Lines of Code	1752				
McCabe Cyclomatic Complexity (avg/max per metho		1.077	0.583	8	/MusicService/src
java		1.083	0.605	8	/MusicService/src
musicservice.dao		1.333	1.404	8	/MusicService/src
UserDao.java		3	2.915	8	/MusicService/src
Tutorial_PageDao.java		1	0	1	/MusicService/src
Tutorial_PageDao		1	0	1	/MusicService/src
findById	1				
findAll	1				
findByTutorialId	1				
TutorialDao.java		1	0	1	/MusicService/src
Tutorial_Page_ResponsesDao.java		1	0	1	/MusicService/src
DemoDao.java		1	0	1	/MusicService/src
DemoDao		1	0	1	/MusicService/src
findById	1				
findAll	1				
LevelDao.java		1	0	1	/MusicService/src
NoteDao.java		1	0	1	/MusicService/src
QuestionDao.java		1	0	1	/MusicService/src

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

1 Open ✔ 14 Closed Author ▾ Labels ▾ Milestones ▾ Assignee ▾ Sort ▾

- **DB version table not up to date** bug
  #44 opened 18 minutes ago by jk214north
- **Cannot perform clean mvn build using VM development environment** bug 1
  #43 opened 21 minutes ago by jk214north
- **Tutorial api can only handle one response key per tutorial page, needs to be able to handle many.** enhancement 1
  #42 opened 27 minutes ago by jk214north
- **Several questions have wrong answer key** bug 1
  #41 opened 32 minutes ago by jk214north
- **Wrong tutorial page type for level 4 and 7 tutorial pages** bug 1
  #40 opened 3 hours ago by jk214north Nov 14 - 20
- **Fix config to allow http requests to multiple_choice.json on production** 1
  #39 opened 22 hours ago by greenelephant1
- **Get rid of click to continue after last quiz** enhancement 1
  #38 opened 22 hours ago by greenelephant1
- **Add a "congrats/complete" page per tutorial** enhancement 1
  #20 opened 18 days ago by witboston Nov 14 - 20
- **Add a "congrats/complete" page per quiz** enhancement 1
  #19 opened 18 days ago by witboston Nov 14 - 20

Deficiency Tracking

1 Open 14 Closed Author Labels Milestones Assignee Sort

Incorporate a Choice object to handle each option for a multiple choice enhancement

#14 opened 24 days ago by witboston Nov 14 - 20

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

Front End Progress

New Features:

- Improved appearance
- More flexibility in tutorial types
- More flexibility in tutorial responses

Refactoring:

- Refactored UI JS to isolate API calls to make connecting with the back end easier (possible)
- Made code cleaner and better

Back End Progress

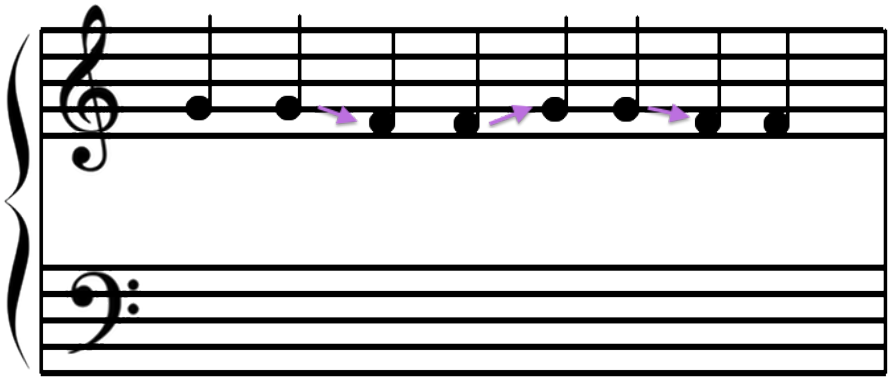
Accomplishments:

- Developed and Deployed REST resources
- Solidified demo, quiz questions & levels in database.
- Built our Azure production environment
  - http://keyboard.cloudapp.net/MusicService
  - Centralizes & Eases development for front-end

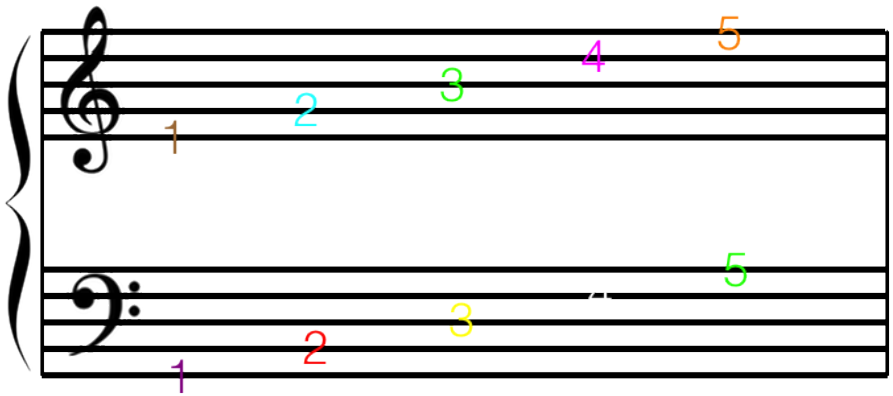
Refactoring:

- Database schema changes to utilize a Level table for demos, tutorials and quizzes.

Tutorial/Quiz Method



Which notes are displayed? Can you play them on the piano?



There are five lines in each stave.
These lines can have notes.

Refactor Examples

and:

to:

and:

Risk Status Update

Risk Category	Risk Name	Risk Number	Probability (1-5)	Impact (1-5)	Orig. Score	Risk Score	Mitigation	Contingency	Time Phase
Team Members	Dropping Class / Losing key members (music) of team mid-stream	4	5	3	6	15	Have a lot of communication between the team members	Remove incomplete capabilities from the project	All
Implementation	Project not completed by deadline.	1	2	5	10	10	Determine what the most basic viable version for this app is and building that as soon as possible. Then we can spend the rest of the semester adding as many features as we have time for.	Demosnstrate incomplete project as best as possible	Fianal Execution
Implementation	Application Performance inhibits usefulness of application	9	1	5	10	5	Prototype shall be produced to determine best architecture/design approach.	Demosnstrate non-functional project as best as possible	Prototyping
Team Members	Slow progress or bad code due to inexperienced team members	2	2	3	9	6	1) Break down the project into smaller more easily manageable pieces, 2) having code reviews and 3) having a lot of communication between the team to help people get up to speed.	Remove incomplete capabilities from the project	All
Implementation	Complexity of working with sound within JavaScript	7	1	4	8	4	Prototype shall be produced to try to flush out potential issues to ensure sounds generated are correct for a keyboard.	Change Strategy	Prototyping

Software Quality Metrics

- Defects
  - Use gitHub issue tracker to track defects
  - Start tracking defects after component has been pushed to development branch
  - Dividing resolved known defects by known defects
  - Defect density by dividing known defects by the number of lines of code
- McCabe’s Cyclomatic Complexity
  - Sections of code with no branches
  - Determine the number of tests required to obtain complete coverage.
  - Indicate the psychological complexity of a method.
- Efferent Couplings
  - Measure of the number of types the class being measured 'knows' about.
  - A large efferent coupling can indicate that a class is unfocussed and also may indicate brittleness

Software Quality Metrics

- Lack of Cohesion
  - Cohesion is an important concept in OO programming.
  - It indicates whether a class represents a single abstraction or multiple abstractions.
  - if a class represents more than one abstraction, it should be refactored into more than one class,
- Lines Of Code
  - This measure indicates the number of lines a method occupies
  - Useful for future estimating

Lines of Code

oblems | Javadoc | Declaration | Diagrams | Metrics - Metrics

etric	Tota	Mean	Std. De	Maximu	Resource causing Maximum	Metho
Total Lines of Code	640					
src	640					
musicservice.model	292					
User.java	111					
Quiz.java	67					
Demo.java	57					
Level.java	57					
musicservice.dao	176					
musicservice.rest	154					
musicservice.common	12					
musicservice.common.constants	6					
McCabe Cyclomatic Complexity (avg/max		1.131	0.896	8	/Metrics/src/musicservice/dao/UserDao.java	regist
Efferent Coupling (avg/max per package		2.8	1.833	5	/Metrics/src/musicservice/dao	
Afferent Coupling (avg/max per package		4.4	3.611	9	/Metrics/src/musicservice/common	
Lack of Cohesion of Methods (avg/max p		0.201	0.336	0.875	/Metrics/src/musicservice/model/User.java	
Method Lines of Code (avg/max per met	277	4.541	6.769	43	/Metrics/src/musicservice/dao/UserDao.java	regist
Number of Methods (avg/max per type)	60	4	4.115	17	/Metrics/src/musicservice/model/User.java	

McCabe's Cyclomatic Complexity

Problems | Javadoc | Declaration | Diagrams | Metrics - Metrics

Metric	Tota	Mean	Std. De	Maximu	Resource causing Maximum	Method
Total Lines of Code	640					
McCabe Cyclomatic Complexity (avg/max		1.131	0.896	8	/Metrics/src/musicservice/dao/UserDao.java	register
src		1.131	0.896	8	/Metrics/src/musicservice/dao/UserDao.java	register
musicservice.dao		1.8	2.088	8	/Metrics/src/musicservice/dao/UserDao.java	register
UserDao.java		3	2.915	8	/Metrics/src/musicservice/dao/UserDao.java	register
UserDao		3	2.915	8	/Metrics/src/musicservice/dao/UserDao.java	register
register	8					
delete	2					
findById	1					
findAll	1					
DemoDao.java		1	0	1	/Metrics/src/musicservice/dao/DemoDao.java	findById
QuizDao.java		1	0	1	/Metrics/src/musicservice/dao/QuizDao.java	findById
LevelDao.java		1	0	1	/Metrics/src/musicservice/dao/LevelDao.java	findById
musicservice.common		1	0	1	/Metrics/src/musicservice/common/LoggerUtils.	getLogger
musicservice.model		1	0	1	/Metrics/src/musicservice/model/Quiz.java	getQuizId
musicservice.rest		1	0	1	/Metrics/src/musicservice/rest/RestController.ja	getKeys
musicservice.common.constants		0	0			
Efferent Coupling (avg/max per package		2.8	1.833	5	/Metrics/src/musicservice/dao	
Afferent Coupling (avg/max per package		4.4	3.611	9	/Metrics/src/musicservice/common	
Lack of Cohesion of Methods (avg/max p		0.201	0.336	0.875	/Metrics/src/musicservice/model/User.java	
Method Lines of Code (avg/max per met	277	4.541	6.769	43	/Metrics/src/musicservice/dao/UserDao.java	register
Number of Methods (avg/max per type)	60	4	4.115	17	/Metrics/src/musicservice/model/User.java	

Lack of Cohesion

Problems | Javadoc | Declaration | Diagrams | Metrics - Metrics

Metric	Tota	Mean	Std. De	Maximu	Resource causing Maximum	Method
Total Lines of Code	640					
McCabe Cyclomatic Complexity (avg/max		1.131	0.896	8	/Metrics/src/musicservice/dao/UserDao.java	register
Efferent Coupling (avg/max per package		2.8	1.833	5	/Metrics/src/musicservice/dao	
Afferent Coupling (avg/max per package		4.4	3.611	9	/Metrics/src/musicservice/common	
Lack of Cohesion of Methods (avg/max p		0.201	0.336	0.875	/Metrics/src/musicservice/model/User.java	
src		0.201	0.336	0.875	/Metrics/src/musicservice/model/User.java	
musicservice.model		0.752	0.089	0.875	/Metrics/src/musicservice/model/User.java	
musicservice.common		0	0	0	/Metrics/src/musicservice/common/LoggerUtils.	
musicservice.common.constants		0	0	0	/Metrics/src/musicservice/common/constants/C	
musicservice.dao		0	0	0	/Metrics/src/musicservice/dao/UserDao.java	
musicservice.rest		0	0	0	/Metrics/src/musicservice/rest/RestController.ja	
Method Lines of Code (avg/max per met	277	4.541	6.769	43	/Metrics/src/musicservice/dao/UserDao.java	register
Number of Methods (avg/max per type)	60	4	4.115	17	/Metrics/src/musicservice/model/User.java	
Nested Block Depth (avg/max per metho		1.066	0.4	4	/Metrics/src/musicservice/dao/UserDao.java	register
Depth of Inheritance Tree (avg/max per		1.067	0.249	2	/Metrics/src/musicservice/common/LoggerUtils.	
Number of Packages	5					
Number of Interfaces (avg/max per pack	0	0	0	0	/Metrics/src/musicservice/common	

JavaScript Sample

jsmeter.info

Edit | Visualize

Line	Function	Statements	Lines	Comment Lines	Comment%	Branches	Depth	Cyclomatic Complexity
1	[[code]]	343	575	15	2.61%	0	0	1
1	(Anonymous1)	6	2	0	0%	0	0	1
2	(Anonymous1).level_overview	5	1	0	0%	0	0	2
5	(Anonymous2)	165	286	8	2.8%	0	0	1
6	(Anonymous2).tutorial_data	164	285	8	2.81%	2	2	5
299	(Anonymous3)	163	277	7	2.53%	0	0	1
300	(Anonymous3).quiz_data	162	276	7	2.54%	2	2	5

Productivity So Far

- Front End:
  - LOC = 809
  - LOC/Hour = 5.15
  - Does not include Angular or JQuery java Script
- Back End:
  - LOC = 640
  - LOC/Hour = 8.48

Lessons Learned

- Coordination is hard (but easier in person)
- Code reviews are awesome
- Programming is easy. Programming in a clear and idiomatic way is hard!
- We should do more structured testing
- Learning curve setting up Azure cloud environment

Next Phase

- Tie front-end and backend components together.
- Focus on testing and QA
- Make bug fixes as needed.

The End!

THANK YOU!

Features & Motivation

Motivation/Adjustments:

- Tutorial/Game to learn how to read and play musical notes
- Train ear to associate notes with a value (C4, E, etc…)
- “Free Play” keyboard online

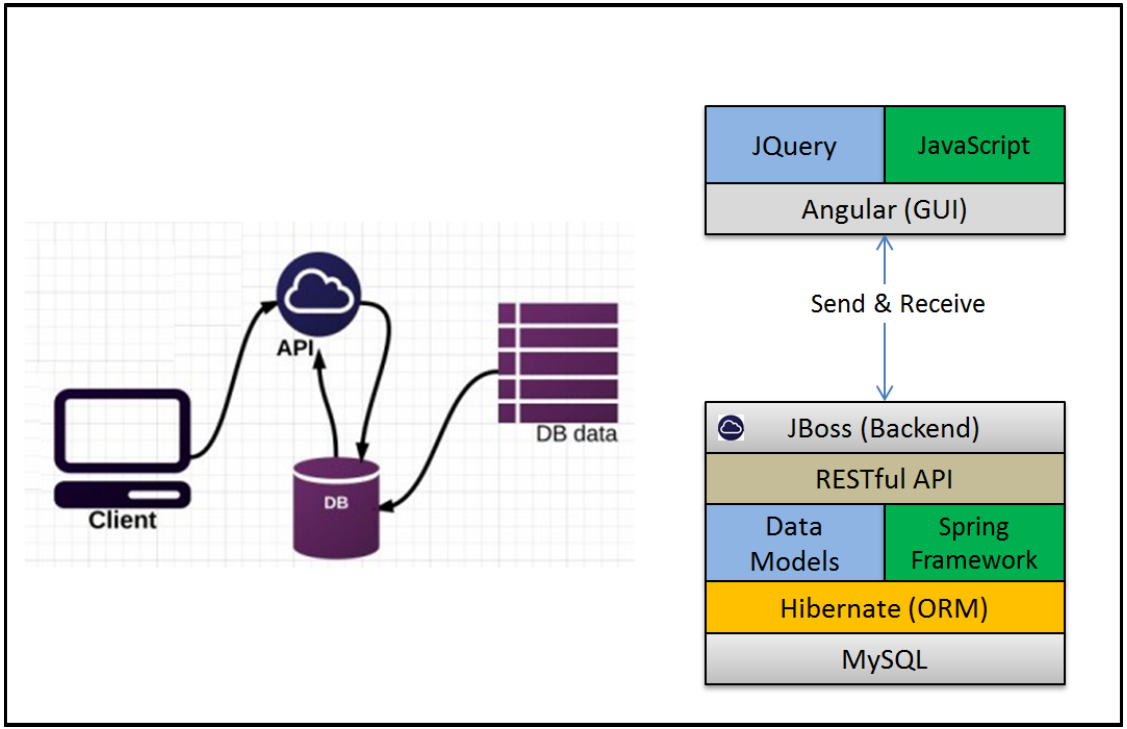
Features:

- Web based Front-end with RESTful Service on back-end
- Progress through tutorials and learn simple songs
- As user completes levels, they will unlock more challenging levels
- Save scores and progress of each lesson

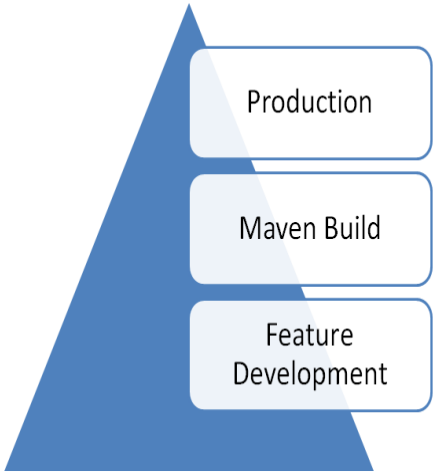
Team Progress

- Freeplay and Tutorial proof of concept.
- Base development complete for:
    - RESTful Spring functionality
    - Database Hibernate functionality
    - Spring Data Models
- Cloned/shared development environment
- Investigated GUI APIs to utilize (Angular, etc…)

High-Level Architecture



Release Process

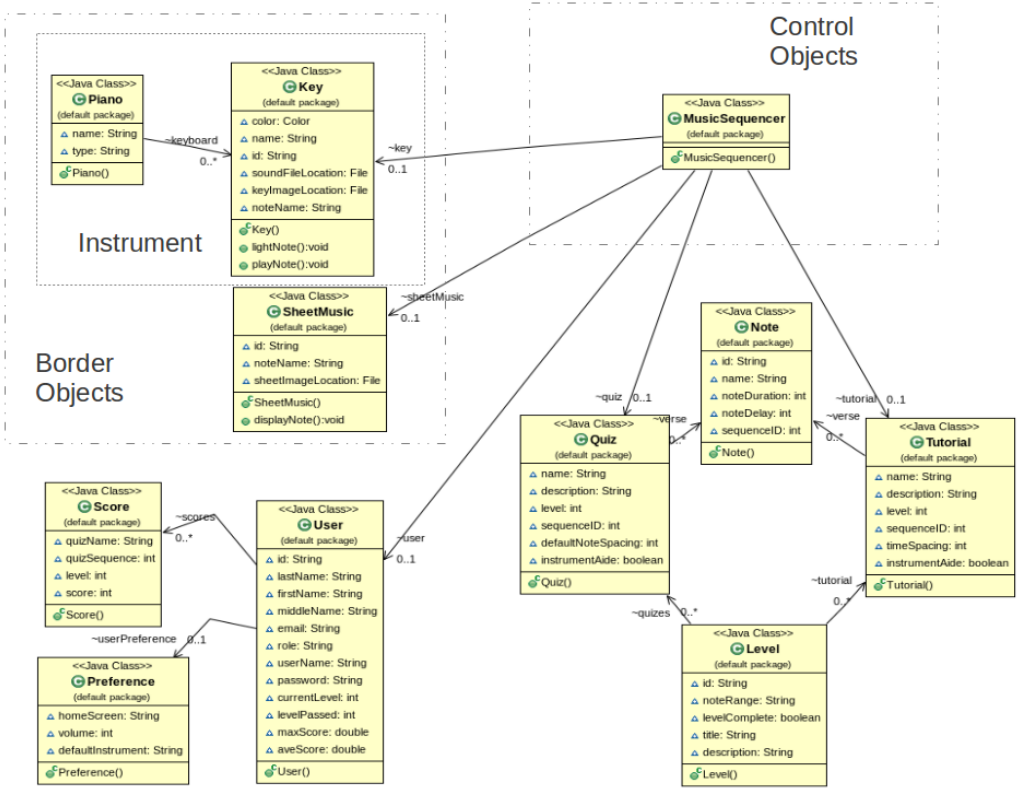


Schedule & Monitoring

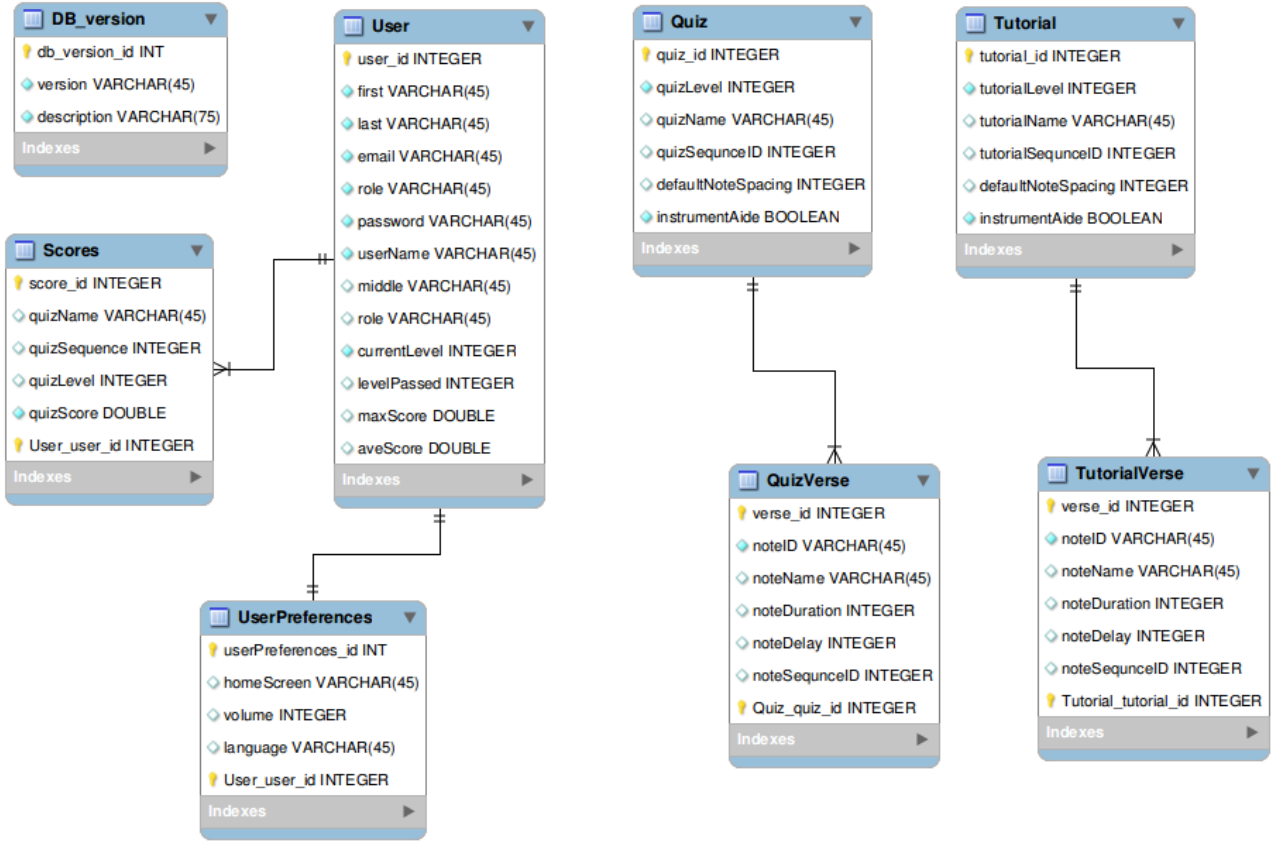
Delivery Schedule

Phase	Start	Finish	Actual	Status
Milestone 1	09/26/2014	10/09/2014		Design Phase
Milestone 2	10/10/2014	10/23/2014		Develop base framework for front & backend
Milestone 3	10/24/2014	11/07/2014		Develop independent components for front and backend
Milestone 4	11/08/2014	11/21/2014		Start Integrating frontend and backend connectivity
Milestone 5	11/22/2014	12/05/2014		Solidify required functionality & minor enhancements

Object/Data Model



DB/Table Model



Project Components - GUI View

Project Components - UI Continued

- Use REST api calls to interact with the back end
  - Front end in entirely separate code base
  - HTTP Verbs: GET, PUT, POST, DELETE
- Use angularJS to update views. The main advantages of angular for this project are
  - Two way data binding
  - Templates
  -

Demos

The End!

THANK YOU!

Risk Management Plan

- Key Elements prior to executing Plan
  - Define work scope, schedule, resources, and cost elements
  - Define minimum and maximum baseline thresholds
  - Define Risk Management Roles and Responsibilities
- Identify Risks
  - Each team member submits top 2 or 3
- Rank Risks
  - Probability (1-5)
  - Operational Impact (1-5)
- Mitigation Strategy
  - Med and high risks only
- Contingency Strategy

Coding Standards

- Why?
  - Greater consistency between developers
  - Easier to understand
  - Easier to maintain and develop
  - Promotes code reuse vs. scrap and re-write
- Industry Standards
  - Comments
  - Naming Convention
  - White Space
- 4 Sections
  - JAVA, JavaScript, HTML, and MySQL

Quality

- Verify acceptance criteria per user story (demo)
- Build tests for each functional component developed
- Use Maven to build and deploy
- Develop central logging to simplify debugging
- Design RESTful data packets to simplify parsing
  - less code paths