ICS673F14 Software Engineering
Group Project 2 - Keyboard Application
Software Design Document

Your project Logo here if any

Team Member	Role(s)	Signature	Date
Nigel Stuart Lead	• Section 1 - Introduction • Section 2 - Software Architecture (Service) • Section 3 - Design Patterns (Service)	*Nigel Stuart*	10/15/2014
Ana Beglova Configuration Leader	• Section 2 - Software Architecture (UI) • Section 3 - Design Patterns (UI)		
Jonathan Kelley Backup Team Lead Implementation Lead	• Section 5		
Christopher Wright Design	• Section 4 - Key Algorithms		

Revision History:

Version	Author	Date	Change
1.0	Christopher Wright	10/9/14	Introduction Completed
1.1	Nigel Stuart	10/15/2014	Added RESTful API design items
1.2	Jonathan Kelley	10/15/2014	Added Current UML and Database Model diagrams

Contents

1. Introduction

1.1 Purpose

This document describes the architecture, methods, and system design used to build a keyboard web application. The application is being designed to be used as a tool to instruct users with little to no musical background how to read sheet-music notation and where to find those notes on a piano keyboard.

1.2 Aim

This document serves many purposes:

- Outline the structures, methods, and approach to the design of the keyboard project
- Track updates to the application and observe their impact on requirements
- Describe resources and tools used in building the project

1.3 **Scope**

As stated above, the SDD outlines the aspects of engineering applied to building a piano-web application. It is intended for students of any age who wish to learn the basics of note reading and apply those fundamentals to a piano keyboard. This tool can be used by students and instructors as part of lesson activities. The goal is to have an application which presents a note reading method in the form of tutorials and games. The benefits of this project include reinforced note and sight reading skills while keeping the students engaged. The tutorials apply mostly to a piano interface, but can be used as a note reading tool for many additional instruments.

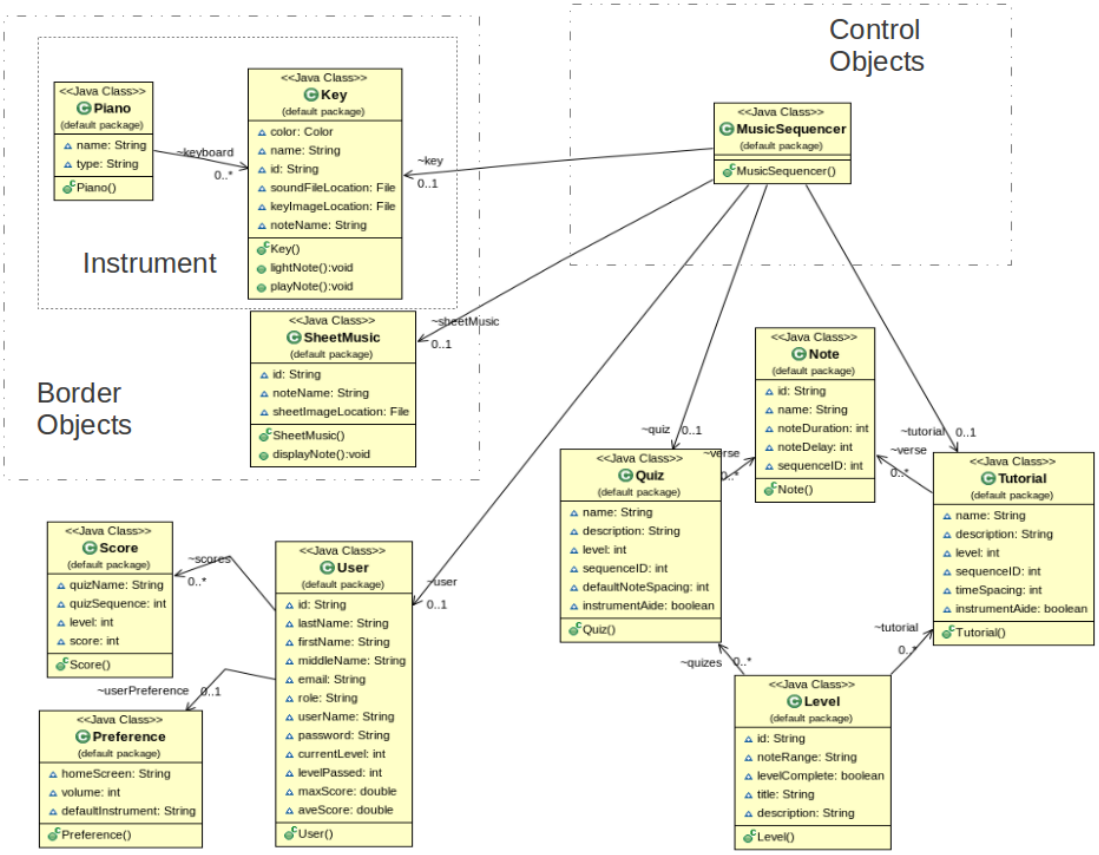
2. Software Architecture

In this section, you will describe the decomposition of your software system, which include each component (which may be in terms of package or folder) and the relationship between components. You shall have a diagram to show the whole architecture, and class diagram for each component. The interface of each component and dependency between components should also be described. If any framework is used, it shall be defined here too. Database design should also be described if used.

a. Software Components
b. Architecture

Architecture Diagram

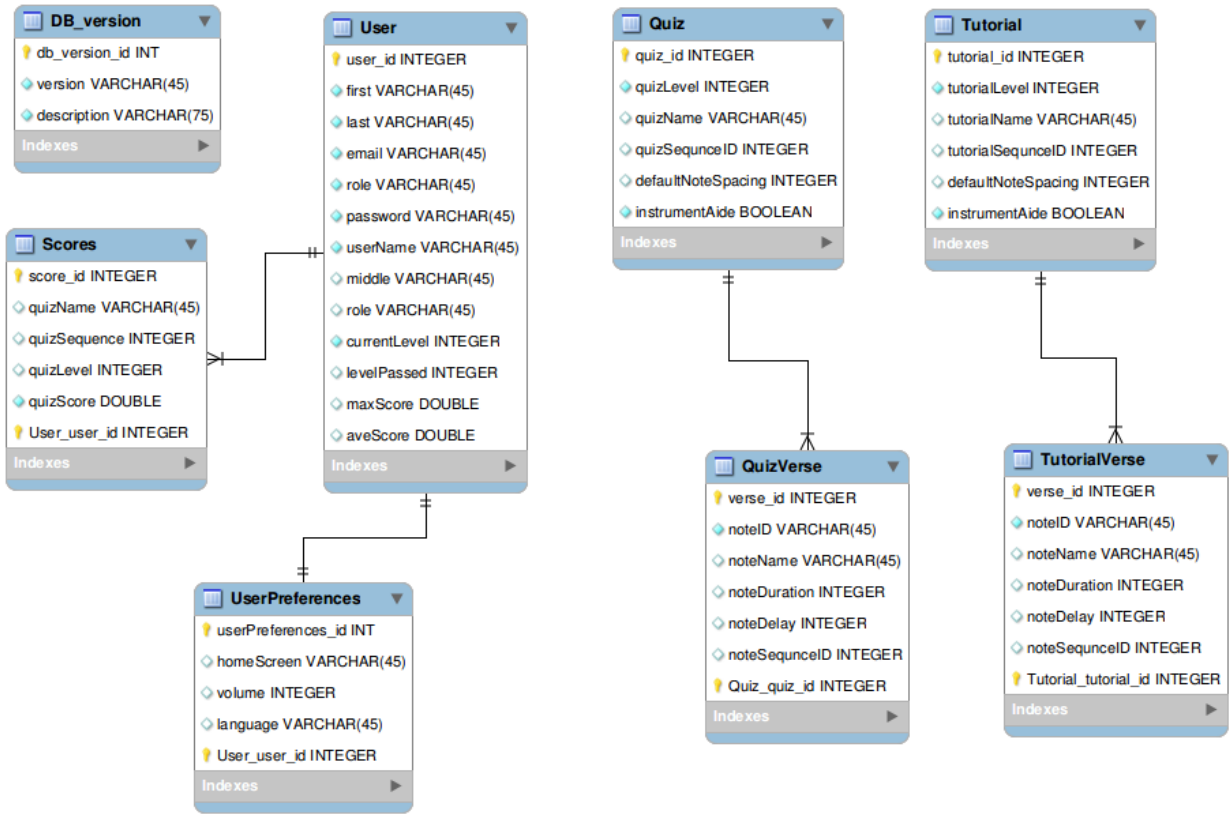
i. Class Diagram - The current UML Class Diagram is shown below. Updates will be made after each increment milestone.



c. Component Interfaces

Framework

d. Database Model



3. Design Patterns

In this section, you shall describe any design patterns used in your software system.

4. Key Algorithms

In this section, you shall describe any key algorithms used in your software system, either in terms of pseudocode or flowchart.

5. Classes and Methods

a. The current UML Class Diagram is shown below in figure a-1. Updates will be made after each increment milestone.

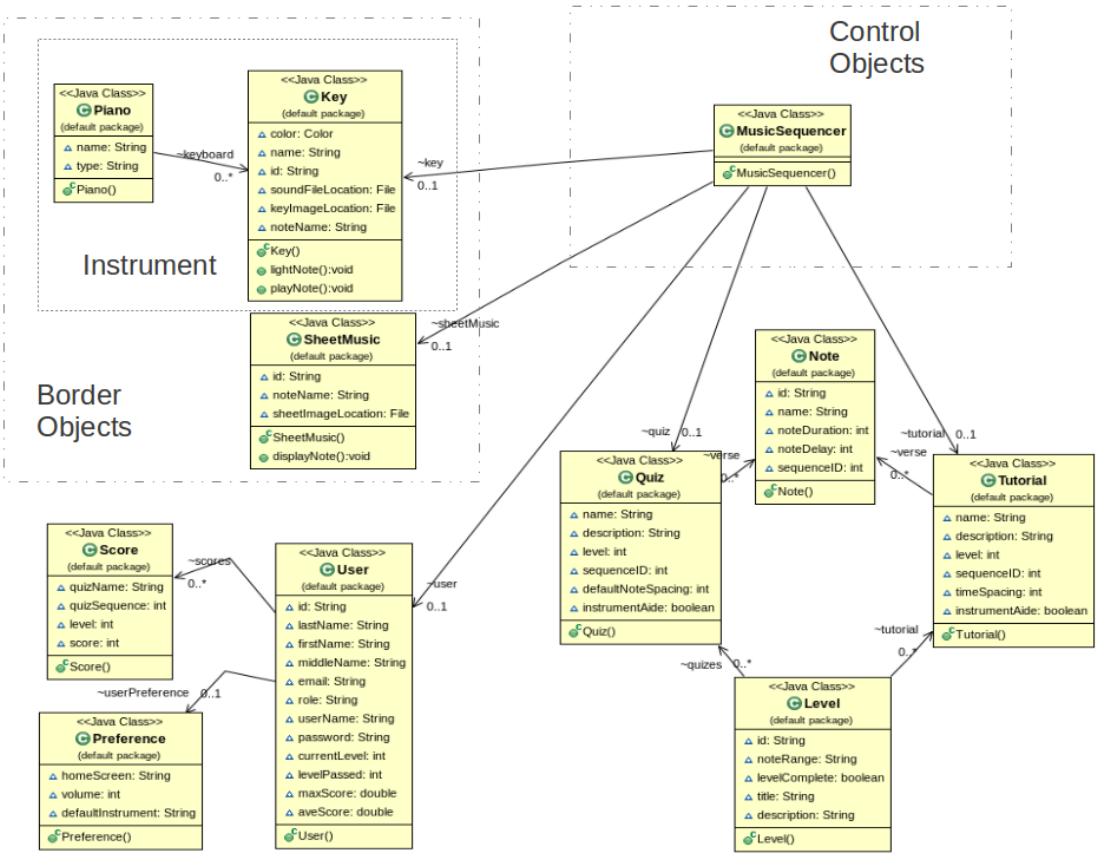


Figure a-1

6. RESTful API

/user

This resource is used to manipulate user data. User data will contain information about each user of the application.

GET /user

Description:
This method will return information about all users in the database. Filters can be applied to limit the results. See inputs section for details about the filters that can be supplied.

Input:

username - The username of the user.
role - Get the users which match the role supplied

Returns:

```
{
   "user": [
     {
       "-username": "pianoMan26",
       "-id": "1"
       "-role": "user"
     },
     {
       "-username": "rockStar",
       "-id": "2"
       "-role": "admin"
     },
     {
       "-username": "JamsterJenny",
       "-id": "3"
       "-role": "user"
     }
   ]
}
```

Example:

http://localhost:8080/MusicService/user

http://localhost:8080/MusicService/user?username=pianoMan26

POST /user

Description:

This method will insert a new user into the database. If a user already exists with the same email address, an error will be thrown.

Input:

{
“username” : <username>,
“role” : <*role*>,
“email” : <*email*>,

“first” : <*first_name*>,
“last : <*last_name*>
}

Returns:

id - The ID of the newly inserted user.

Example:

http://localhost:8080/MusicService/user

/tutorial

This resource is used to manipulate tutorial data.

GET /tutorial?username=<id>

Description:

This method will return high level information about all tutorials for a specific user. It will provide information about the status of the user’s progress per tutorial. If a user has completed a tutorial its status will be “Complete”. If a user has started a tutorial, its status will be “Unlocked”. If a tutorial has not been reached yet, because prior levels have not been completed, the status for that level will be “Locked”.

Input:

None

Returns:

```
{
   "tutorial": [
     {
       "-name": "level 1",
       "-id": "1"
       "-status": "Complete"
     },
     {
       "-name": "level 2",
       "-id": "2"
       "-status": "Complete"
     },
     {
```

```
      "-name": "level 3",
      "-id": "3"
      "-status": "Unlocked"
    },
    {
      "-name": "level 4",
      "-id": "4"
      "-status": "Locked"
    },
    {
      "-name": "level 5",
      "-id": "5"
      "-status": "Locked"
    },
    {
      "-name": "level 6",
      "-id": "6"
      "-status": "Locked"
    }
  ]
 }
}
```


Example:

http://localhost:8080/MusicService/tutorial?username=856

GET /tutorial/<id>

Description:

This method will return details about a specific tutorial level. These details will not be user specific. They will only contain metadata about the tutorial.

Input:

None

Returns:


```
{
 "entries": {
   "level": [
     {
       "-name": "level 1",
       "-id": "1",
       "-description": "This tutorial will help you learn notes C4 and E",
```

```
       "question" : [

                    ]
      },
   ]
 }
}
```

Example:

/quiz

This resource is used to manipulate quiz data.

GET /quiz/user/<id>

Description:

This method will return high level information about all quizzes for a specific user. It will provide information about the status of the user's progress per quiz. If a user has completed a quiz its status will be "Passed". If a user has started a quiz, its status will be "Started". If a tutorial has not been started yet, because the tutorial has not been completed, then the status will be "Locked".

Input:

None

Returns:

```
{
 "entries": {
   "quiz": [
     {
       "-name": "Quiz 1",
       "-id": "1"
       "-status": "Passed"
     },
     {
       "-name": "Quiz  2",
       "-id": "2"
       "-status": "Passed"
     },
     {
       "-name": "Quiz 3",
```

```
            "-id": "3"
            "-status": "Started"
          },
          {
            "-name": "Quiz 4",
            "-id": "4"
            "-status": "Locked"
          },
          {
            "-name": "Quiz 5",
            "-id": "5"
            "-status": "Locked"
          },
          {
            "-name": "Quiz 6",
            "-id": "6"
            "-status": "Locked"
          }
        ]
      }
    }
```


Example:

http://localhost:8080/MusicService/tutorial/6

GET /quiz/<id>

Description:

This method will return details about a specific quiz level.

Input:

None

Returns:


```
{
 "entries": {
   "question": [
     {
       "-name": "Question 1",
       "-id": "1",
       "-question": "What hand do you use to play this note?",
       "-type": "Multiple_Choice",
       "-image": "http://localhost/images/quiz_1_q1.jpg",
```

```
      "-answer": "Right",
      "-":options": [
            {"-display": "Left"},
            {"-display": "Right"}
    },
    {
      "-name": "Question 2",
      "-id": "1",
      "-question": "What hand is Shawn?",
     "-type": "Multiple_Choice",
     "-image": "http://localhost/images/quiz_1_q2.jpg",
     "-answer": "Right",
      "-":options": [
            {"-display": "Left"},
            {"-display": "Right"}
    },
    {
      "-name": "Question 3",
      "-id": "1",
      "-question": "Which key is highlighted? ",
     "-type": "Multiple_Choice",
     "-image": "http://localhost/images/quiz_1_q2.jpg",
     "-answer": "A",
      "-":options": [
            {"-display": "A"},
          {"-display": "B"}
          {"-display": "C"}
          {"-display": "D"}
    },
  ]
 }
}
```

PUT /quiz/user/<id>

Description:
Input:
None
Returns:

Example:
http://localhost:8080/MusicService/tutorial/6

PUT /instance/quiz/

Description:
Use this resource to save a users score for a quiz to the database
Input:
score - the score [PASS/FAIL] quiz_id - the quiz metadata id user_id - The id of the user in the database
Returns:
Example:

Tutorial Section

/level

This resource is used to retrieve level information data.

GET /level

Description:
This method will return high level information about all levels. It will provide the name and description of all available levels
Input:
None
Returns:


```
{
"level": [
  {
  "-name": "level 1",
  "-id": "1",
  "-description": "This is the easiest and first level"
  },
  {
  "-name": "level 2",
  "-id": "2",
  "-description": "This is the 2nd to easiest and second level"
  },
  {
  "-name": "level 3",
  "-id": "3",
  "-description": "This is the 3rd to easiest and third level"
  },
  {
  "-name": "level 4",
  "-id": "4",
  "-description": "This is the 4th to easiest and fourth level"
  },
  {
  "-name": "level 5",
  "-id": "5",
```

```
  "-description": "This is the 5th to easiest and fifth level"
  },
  {
  "-name": "level 6",
  "-id": "6",
  "-description": "This is the 6th to easiest and sixth level"
  }
]
}
}
```


Example:

http://localhost:8080/MusicService/level

/tutorial?levelID=<id>

This method will return high level information about all turorials for a certain level.

GET /tutorial?levelID=<id>

Description:
This method will return high level information about all turorials for a certain level. It will provide the ID, sequence ID, name and description of all available tutorials for a level.
Input:
None
Returns:

```
{
 "tutorial": [
  {
  "-name": "tutorial 2-1",
  "-id": "10",
  “-sequence-ID” : “1”,
  "-description": "This is the first tutorial for the second level",
  “-type” : “Question_Answer”
  },
  {
  "-name": "tutorial 2-2",
  "-id": "11",
  “-sequence-ID” : “2”,
  "-description": "This is the 2nd tutorial for the second level",
  “-type” : “Press_Key”
  },
  {
  "-name": "tutorial 2-3",
  "-id": "12",
  “-sequence-ID” : “3”,
  "-description": "This is the 3rd tutorial for the second level",
  “-type” : “Press_Key”
  },
  {
  "-name": "tutorial 2-4",
  "-id": "13",
```

```
  “-sequence-ID” : “4”,
  "-description": "This is the 4th tutorial for the second level",
  “-type” : “Player_Piano”
  }
  ]
 }
}
```


Example:

http://localhost:8080/MusicService/tutorial?levelID=2

/tutorial?tutorialID=<id>

This method will return all information about a specific turorials for a specific ID.

GET /tutorial?tutorialID=<id>

Description:
This method will return all information about a specific turorials for a specific ID. It will provide all the attributes of a specific tutorial.
Input:
None
Returns:

```
Example 1:
{
 "tutorial":
  {
  "-name": "tutorial 2-1",
  "-id": "10",
  “-sequence-ID” : “1”,
  "-description": "This is the 1st tutorial for the second level",
  “-type” : “Question_Answer”,
  “-instrumentAide” : “true”,
  “-scriptItems” : [
     {
      “id” : “33”,
      “name” : “question 1”,
      “sequenceID” : “1”,
      “questionType” : “multiple_choice”,
      “questionText” : “please identify the correct note: A) e#; B) c1; C)
f!; D) c2.”,
      “questionImage” : ”c1”,
      “expectedResponse” : “B”,
      “associatedNote” : “”,
      “noteDuration” : “”,
      “noteDelay” : “”
      },
      {
```

```
      “id” : “34”,
      “name” : “question 2”,
      “sequenceID” : “2”,
      “questionType” : “informational”,
      “questionText” : “this would be the paragraph for left and right
handed play”,
      “questionImage” : ””,
      “expectedResponse” : “continue”,
      “associatedNote” : “”,
      “noteDuration” : “”,
      “noteDelay” : “”
      },
  ]
  }
}

Example 2:
{
 "tutorial":
  {
  "-name": "tutorial 2-4",
  "-id": "13",
  “-sequence-ID” : “4”,
  "-description": "This is the 3rd tutorial for the second level and
contains all the information to play Happy Birthday",
  “-type” : “Player_Piano”,
  “-instrumentAide” : “true”,
  “-scriptItems” : [
     {
      “id” : “133”,
      “name” : “note 1”,
      “sequenceID” : “1”,
      “questionType” : “”,
      “questionText” : “”,
      “questionImage” : ”c1”,
      “expectedResponse” : “”,
      “associatedNote” : “c1”,
      “noteDuration” : “1190”,
      “noteDelay” : “90”
      },
      {
      “id” : “134”,
      “name” : “note 2”,
      “sequenceID” : “2”,
      “questionType” : “”,
      “questionText” : “”,
      “questionImage” : ”d1”,
      “expectedResponse” : “”,
```

```
      "associatedNote" : "d1",
      "noteDuration" : "2190",
      "noteDelay" : "80"
      },
      {
      "id" : "135",
      "name" : "note 3",
      "sequenceID" : "3",
      "questionType" : "",
      "questionText" : "",
      "questionImage" : "f1#",
      "expectedResponse" : "",
      "associatedNote" : "f1#",
      "noteDuration" : "1000",
      "noteDelay" : "500"
      },
      {
      "id" : "136",
      "name" : "note 4",
      "sequenceID" : "4",
      "questionType" : "",
      "questionText" : "",
      "questionImage" : "d1",
      "expectedResponse" : "",
      "associatedNote" : "d1",
      "noteDuration" : "1000",
      "noteDelay" : "100"
      },
  ]
  }
}
```


Example:

http://localhost:8080/MusicService/tutorial?tutorialID=11
http://localhost:8080/MusicService/tutorial?tutorialID=12

7. References

8. Glossary