

University of Nevada, Reno

**ARIA: Administration, Registration, and Information Assistant**

A thesis submitted in partial fulfillment  
of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering and the Honors Program

by

**Renee T. Inuma**

Wesley Kepke

Ernest Landrito

Kyle Lee

Dr. Frederick C. Harris, Jr., Thesis Advisor

May, 2016

**UNIVERSITY  
OF NEVADA  
RENO**

**THE HONORS PROGRAM**

We recommend that the thesis  
prepared under our supervision by

**RENEE T. IINUMA**

entitled

**ARIA: Administration, Registration, and Information Assistant**

be accepted in partial fulfillment of the  
requirements for the degree of

**BACHELOR OF SCIENCE, COMPUTER SCIENCE AND ENGINEERING**

---

Frederick C. Harris, Jr., Ph.D., Thesis Advisor

---

Tamara Valentine, Ph.D., Director, Honors Program

May, 2016

*ABSTRACT*

---

Administration, Registration, and Information Assistant (ARIA) is a WordPress plugin that assists the Northern Nevada Music Teachers Association (NNMTA) with the organization and management of their annual music festivals and competitions. This project significantly reduces the amount of time that the NNMTA staff spends in preparation for their music competitions. The plugin automates tasks that would otherwise take NNMTA members hours to complete; the three main components of ARIA include registration, scheduling, and document generation for every competition. With the completion of ARIA, the NNMTA now has a robust and systematic manner of hosting music competition for years to come.

*ACKNOWLEDGEMENTS*

---

Renee would like to thank her teammates: Wesley Kepke, Ernest Landrito, and Kyle Lee. She would also like to thank Dr. Fred Harris (CSE), Mrs. Cindy Harris (NNMTA), and Dr. Sergiu Dascalu (CSE) for their guidance and support of this project.

*TABLE OF CONTENTS*


---

<b>Introduction .....</b>	<b>1</b>
<b>1 Project Concept .....</b>	<b>2</b>
1.1 Overview .....	2
1.2 Significance .....	3
1.2.1 Similar Applications and References .....	4
1.2.2 Innovation.....	5
<b>2 Specification.....</b>	<b>6</b>
2.1 Project Management.....	6
2.1.1 Project Components.....	6
2.1.2 Team Members .....	8
2.2 Requirements .....	10
2.2.1 Requirements Workshop Summary .....	10
2.2.2 Software Requirements .....	12
2.3 Use Case Modeling .....	15
2.3.1 Use Case Diagram .....	16
2.3.2 Advanced Use Case Diagrams .....	17
2.3.4 Detailed Use Cases .....	20
2.4 Requirement Traceability Matrix.....	23
<b>3 Analysis .....</b>	<b>24</b>
3.1 Analysis Diagrams .....	24
3.2 Analysis Packages.....	25
<b>4 Design and Implementation.....</b>	<b>28</b>
4.1 Design Model.....	28
4.1.1 Architectural Design .....	28
4.2 Class Diagrams .....	30
4.3 Program Units .....	37
4.4 Detailed Design.....	49
4.5 Data Design.....	54
4.6 User Interface Design.....	57
<b>Conclusion.....</b>	<b>70</b>
<b>Bibliography .....</b>	<b>71</b>
<b>Appendix A – Glossary of Terms .....</b>	<b>72</b>

*LIST OF TABLES*

---

Table 1: Summary of functional and non-functional requirements .....	14
Table 2: View Teacher Form use case details.....	21
Table 3: Upload Music Info use case details. ....	21
Table 4: Pay Competition Fee use case details. ....	22
Table 5: Create Competition use case details. ....	22
Table 6: Requirements Traceability Matrix .....	23

*LIST OF FIGURES*

---

Figure 1: Diagram of project use cases.....	16
Figure 2: Advanced Use Case 1. Competition Creation Use Cases.....	17
Figure 3 Advanced Use Case 2. Music Info inclusion. ....	17
Figure 4: Analysis model for ARIA.. ....	24
Figure 5: Data manager analysis package.....	25
Figure 6: Competition registration analysis package. ....	26
Figure 7: Competition scheduler package .....	26
Figure 8: Competition document manager package .....	27
Figure 9: High-level structural diagram of ARIA. ....	29
Figure 10: High-level activity flow. ....	30
Figure 11: GUI design class.....	31
Figure 12: Scheduler design class .....	33
Figure 13: Competition design class. ....	34
Figure 14: Registration design class. ....	35
Figure 15: Music design class. ....	36
Figure 16: Document design class.....	36
Figure 17: Statistics design class.....	37
Figure 18: Activity diagram for the process of creating a competition.....	50
Figure 19: Activity diagram for the student registration process.....	51
Figure 20: Activity diagram for the teacher registration process.....	52
Figure 21: Activity diagram for the process of document generation.....	53

***LIST OF ILLUSTRATIONS***

---

Illustration 1: Sample competition creation form .....	57
Illustration 2: Sample competition creation form date selection .....	58
Illustration 3: Sample teacher information upload .....	59
Illustration 4: Sample time entering for competition creation .....	59
Illustration 5: Sample form publication.....	60
Illustration 6: Sample student registration form .....	61
Illustration 7: Sample teacher selection for student registration.....	62
Illustration 8: Sample email to teachers .....	63
Illustration 9: Sample page of incorrect URL.....	63
Illustration 10: Sample teacher registration page.....	64
Illustration 11: Sample period selection .....	65
Illustration 12 and 13: Sample composer selections .....	66
Illustration 14: Sample song selection.....	67
Illustration 15: Sample schedule .....	68
Illustration 16: Sample document.....	69

## *INTRODUCTION*

---

The Northern Nevada Music Teachers Association (NNMTA) is an organization comprised of students and teachers who share a common interest in music. The NNMTA provides opportunities for aspiring musicians and music aficionados by holding competitions, festivals, networking events, and other music-related functions. Despite the diversity of events held by the NNMTA, their annual music competitions are their most prominent affair with interest and enrollment growing each year.

While the board of NNMTA is excited that an expanding number of students are electing to participate in their competitions, the music organization has been experiencing difficulty in accommodating the already abundant body of competitors. Every component of their competition format is facilitated through the use of paper forms, a method prone to several mistakes which can be difficult to alleviate. Parents must register their children by filling out a form by hand and delivering a check in-person to pay. Unfortunately, the current state of the NNMTA's website provides little support for mitigating the organizational problems associated with their music competitions. The NNMTA has been looking for a solution which will allow them to keep up with their growing number of members.

## *1 PROJECT CONCEPT*

---

### **1.1 OVERVIEW**

The main goal of ARIA (Administration, Registration, and Information Assistant) is to modify the NNMTA's website such that the registration, management, and operation of their music competitions can operate in a more efficient and streamlined manner. The intended users fall into three main categories: the NNMTA board members, the music students and their parents, and the music teachers. The NNMTA board members, particularly the music festival chairman, will be using ARIA to create competition forms, schedule competition times, and generate all necessary competition documents. The music students and their parents will be using the forms generated by ARIA to register for competitions, and the music teachers will be using other forms to update the competition information. This project is beneficial to them because ARIA will make music competitions easier to access for all parties. By eliminating the need for paper registration, ARIA eliminates paper waste, while also saving the NNMTA board members time and effort. By moving registration online, festival registration will be easier for parents and music teachers to access, and it will eliminate errors in the process by enforcing registration guidelines automatically.

The main functionality and capabilities of ARIA are to allow competitors to register online, make registration payments, and select a time for their competition. Additionally, ARIA will automate scheduling opportunities to reduce conflicts imposed by paper registration, allow the festival chairman to print out all documents associated with competitions, and even calculate statistics information based on each competition.

ARIA is specifically designed to operate on WordPress; therefore, custom plugins are under development using PHP. Data management and storage is handled in conjunction with the Gravity Forms plugin and API on WordPress. Client-side services are handled with Javascript, jQuery, and AJAX.

## **1.2 SIGNIFICANCE**

ARIA is worthwhile because it has an immediate and real-world impact since it will be used in music competitions occurring annually. ARIA combines a front-end which must be accessible to users, a back-end which must handle dynamic form creation, and the integration of large amounts of data, from the song database to the information of all music students and teachers. These components must work together in a reliable way so that it may be reused and modified as necessary without continued support by the development team.

The potential for further development beyond CS426 includes the option of expanding the functionality of ARIA both specifically for the NNMTA and for the general public. Its potential within NNMTA includes the possibility of adding features and functionality to create and manage music competitions, while the potential for the public includes a generalized system which handles registration and scheduling for any organization or

purpose. Since ARIA can be packaged as an all-purpose registration and scheduling assistant as a WordPress plugin, ARIA is market-viable. Since plugins can be developed and sold for use by the public, there is even the potential for ARIA to become a commercial product.

### **1.2.1 Similar Applications and References**

Similar applications include various plugins listed below:

- RegistrationMagic - Custom Registration Forms
  - This plugin allows the users to create a form with just a few clicks (Registrationmagic, 2015)
- Simple User Registration
  - This allows the administrator to input fields in registration form using a drag and drop meta-page (N-Media, 2014).
- Event Registration
  - It provides registration events, classes, or parties. It is designed to be easy to navigate and allows the user to capture the registering person contact information and any additional information the user request to a database (Marcus).
- Gravity Forms
  - Gravity Forms enables the user to quickly build and design a WordPress form using the form editor ("GravityForms," 2016). The user can select fields, configure options and embed forms on WordPress powered sites. ARIA will be building off of Gravity Forms.

### **1.2.2 Innovation**

The innovative characteristic of the project is the fact that forms are created dynamically for the festival chairman. Upon activation of ARIA, a configuration form is created and published for the user. The festival chairman of NNMTA can then fill out this form (which includes competition name, dates, volunteer times, and other configuration information). When the chairman submits this form, additional (music student and music teacher) forms are automatically generated with all necessary fields, such as name, email, and student level. The chairman does not need to configure each form individually. This process makes the process much easier for the festival chairman. It also sets ARIA apart from other similar plugins because it allows for the possibility of dynamic field creation and population catered to meet the goals of the NNMTA.

## **2.1 PROJECT MANAGEMENT**

### **2.1.1 Project Components**

At a fundamental level, ARIA can be broken down into the following components:

- Music Uploading/Exporting
  - The NNMTA maintains an Excel file that the festival chairman uses to add, delete, and update music pieces that are used in competition. ARIA's music component is responsible for reading this data (imported as a CSV file) and adding all of the entries into a centralized database. Moreover, this component is also responsible for creating a CSV file that be downloaded in the event where the festival chairman does not have access to the original file.
  - Manager: Kyle Lee
- Competition Registration
  - For each competition, registration begins as parents submit information regarding their children. Once student registration has closed, the music teachers of the aforementioned students need to perform additional

registration, both for themselves and also for their students. The successful interaction between these two phases of registration is pivotal.

- Manager: Renee Inuma
- Competition Scheduling
  - After both phases of registration have closed, ARIA will automatically generate scheduling for all students and teachers that have registered for the competition. This component will take into account a substantial amount of information in order to create a schedule that most appropriately fits what the students and teachers had requested.
  - Manager: Wesley Kepke
- Competition Paperwork and Statistics Generation
  - After scheduling is complete, ARIA will provide a mechanism for printing out informational papers, such as which students and teachers are assigned to a competition room at a specific time, that can be used to help NNMTA personnel during competition days. Finally, ARIA will also retain statistics information about a competition, such as the number of students competing and number of teacher-volunteers.
  - Manager: Ernest Landrito

ARIA's developers will be responsible for ensuring that their component can be integrated into the component(s) that precede/succeed the component with which they are directing their focus. While this is not always achievable in practice or occurs without some form of discrepancy, ARIA's developers will strive towards this goal.

A public repository on GitHub ([https://github.com/wesleykepke/ARIA\\_Plugin](https://github.com/wesleykepke/ARIA_Plugin)) is used to store all ARIA source code. Team members will work on files that pertain only to their specific module. The team is also using Slack for all team communication so that team members are kept up-to-date on project module developments.

A potential risk for development includes developers working on the same file at the same time because it is needed to accomplish their module. This can be alleviated by ensuring unambiguous group communication. Continuing, a second risk involves introducing functionality that breaks the current build. This risk can be dismissed by reverting to a prior build on GitHub that functions as expected. Finally, a third risk involves a non-compliant merging of individual modules. As previously mentioned, this can be resolved by working together and discussing what interactions need to take place among modules in order for them to be integrated without errors.

### **2.1.2 Team Members**

All members of ARIA's development team are seniors studying Computer Science & Engineering at the University of Nevada, Reno and are graduating in May 2016.

- Renee Iinuma
  - After spending two years at UC Davis studying chemical engineering, Renee transferred to Nevada and switched her major to computer science. Renee is now the president of ACM at Nevada and works as a tutor at the Engineering Tutoring Center. Her interests include computer graphics, as

well as web applications and front-end development. Renee hopes to work in industry in a field in which she can apply these skills.

- Wesley Kepke
  - At Nevada, Wes enjoyed learning about programming language theory, compiler construction, and computer graphics. While he still maintains an interest in these subjects, Wes would now like to direct his focus towards web and mobile applications and is aiming to land an industry position that pertains to these topics. Wes' beverage of choice while programming is matcha green tea and his favorite programming language is CatfishC, a dialect of C.
- Ernest Landrito
  - Before taking up the keyboard as a Computer Science major, Ernest was a music education major and vocalist. Because of his background, Ernest, is passionate about his work to help the Northern Nevada Music Teachers Association. Ernest is excited to be starting at Google in Mountain View, CA, this upcoming summer, hoping to work on YouTube.
- Kyle Lee
  - Kyle began his education as a mechanical engineer, but switched over to computer science after taking his first computer science course. He currently works at Central Services (Networking) at the university. He is interested in human computer interaction and networking, and his goal is to find a job in these fields.

## **2.2 REQUIREMENTS**

### **2.2.1 Requirements Workshop Summary**

A requirements workshop for ARIA's specification took place with Dr. Harris (UNR CSE), Mrs. Cindy Harris (NNMTA), and all of ARIA's developers on December 21, 2015 at the Harris residence. During this requirements workshop, which lasted approximately one hour, the developers listened intently to the specifications outlined by Mrs. Harris and provided feedback on what seemed achievable given the required completion date. Since Dr. Harris is serving as the technical advisor for ARIA, he also added input regarding the development aspects of ARIA.

Having all six affiliated members at the brainstorming session allowed for many preliminary requirement ambiguities to be resolved and also solidified the requirements desired by the NNMTA. For example, some of the ideas brought up during the brainstorming session were what the required fields for student and teacher registrations were going to be, the functional requirements for the chairman, the requirements necessary for scheduling, and the competition documents that needed to be created.

During the analysis session, the team worked to organize and prioritize the requirements discussed during the brainstorming session. Some of the requirements suggested during the brainstorming session were redundant or very similar, so they were combined or eliminated. For example, it is necessary to upload multiple documents, such as music data or teacher information, to the NNMTA server. These were initially separated into multiple requirements, but the team combined them into one "file upload" requirement. Additionally, the team began prioritizing requirements based on their necessity and

deadlines. For example, the lower division music competition will take place in March, so completing tasks for this competition had the highest priority. Mrs. Harris was most concerned about the process of registration and automating the scheduling for this competition.

The most important finding during this requirements workshop was the shift from a focus on the entire website to a more in-depth service for music festival registration and management. Additionally, other important findings included the need to automate as much of the process as possible, the need to use the Gravity Forms plugin to integrate with ARIA, and the restraints on scheduling for the competitions.

### 2.2.2 Software Requirements

	<b>ID</b>	<b>Requirement</b>
Non-functional Requirements	NR_01	<ul style="list-style-type: none"> <li>• Compatible with all major browsers:               <ul style="list-style-type: none"> <li>○ Since a large volume of users is anticipated, ARIA will need to be compatible with the spectrum of browsers that are being utilized by users. While ARIA’s developers do not anticipate this being a major issue, it is indeed a non-functional requirement that must be adhered to.</li> </ul> </li> </ul>
	NR_02	<ul style="list-style-type: none"> <li>• Developed on the WordPress platform:               <ul style="list-style-type: none"> <li>○ The NNMTA has requested to remain on WordPress; therefore ARIA will be developed using the PHP scripting language (WordPress requires PHP). Additionally, some functionality of ARIA requires client-side processing. To accomplish this, JavaScript will be used to make backend requests and retrieve necessary data.</li> </ul> </li> </ul>
	NR_03	<ul style="list-style-type: none"> <li>• Reasonable response time:               <ul style="list-style-type: none"> <li>○ All operations completed by ARIA should be achieved with a reasonable response time, especially the student and teacher online registration forms.</li> </ul> </li> </ul>
	NR_04	<ul style="list-style-type: none"> <li>• Utilization of Gravity Forms WordPress plugin:               <ul style="list-style-type: none"> <li>○ Creation of registration forms must use the Gravity Forms plugin for form and field creation.</li> </ul> </li> </ul>
	<b>ID</b>	<b>Requirement</b>
Functional Requirements – LEVEL 1	FR_01	<ul style="list-style-type: none"> <li>• Music uploading/downloading:               <ul style="list-style-type: none"> <li>○ ARIA will allow the festival chairman to upload and download NNMTA music. To upload music, the festival chairman will upload a CSV file containing all NNMTA music. ARIA will then process this data and add it to a centralized music database that can be accessed from many points throughout the plugin. If the festival chairman needs a CSV file of all the music, ARIA will generate this file for him/her. Also, if the festival chairman needs to modify the current selection of music, all he/she needs to do is download the CSV file, make changes to this file, and re-upload it.</li> </ul> </li> </ul>

	ID	Requirement
Functional Requirements – LEVEL 1	FR_02	<ul style="list-style-type: none"> <li>• Student and teacher registration:               <ul style="list-style-type: none"> <li>○ ARIA will allow both students and teachers to register for a competition. When a festival chairman creates a competition, two new registration forms will be created, one for the students and another for the teachers. The students will begin by entering their data for a music competition.</li> <li>○ Once this information is submitted, the teacher will receive, via email, a link to a form that has many of the fields already filled out. The data that is filled out will be specific to a student. The teacher then completes a student’s registration by adding some additional information, such as what piece the student will be performing and under what skill level the student is categorized as. The teacher will then submit this form and a student will have been successfully registered for a competition.</li> </ul> </li> </ul>
	FR_03	<ul style="list-style-type: none"> <li>• Scheduling of students and teachers:               <ul style="list-style-type: none"> <li>○ ARIA will perform scheduling of all students on a first-come, first-served basis. Moreover, ARIA will schedule students according to the criteria that the festival chairman defined when creating a competition, such as the amount of students that are allowed per time slot of judging and the duration of the aforementioned time slots. Scheduling is currently one of the most cumbersome activities that NNMTA personnel must undergo; therefore, accomplishing this functional requirement is imperative.</li> </ul> </li> </ul>
	FR_04	<ul style="list-style-type: none"> <li>• Production of competition documents:               <ul style="list-style-type: none"> <li>○ ARIA will produce competition documents, such as which students and teachers are allocated to a competition room at a designated time. This functional requirement is an extension of the scheduling functional requirement and is yet another tedious activity that is currently bothersome to the NNMTA. Generation of these documents will assuredly mitigate problems that occur on competition days.</li> </ul> </li> </ul>

	<b>ID</b>	<b>Requirement</b>
Functional Requirements – LEVEL 2	FR_05	<ul style="list-style-type: none"> <li>• Generation of competition statistics:               <ul style="list-style-type: none"> <li>◦ ARIA should generate statistics information for competitions. This would include, for example, the number of students that are participating under a particular skill level, the amount of teachers that volunteered, and even the amount of superior ratings achieved by students. This information is somewhat utilized for future competitions so producing easy-to-read documents would be of great use to the NNMTA.</li> </ul> </li> </ul>
	FR_06	<ul style="list-style-type: none"> <li>• Support for multiple competitions:               <ul style="list-style-type: none"> <li>◦ ARIA should be capable of supporting multiple competitions that are occurring simultaneously. That is, students and teachers should be able to sign up for a specific competition in which the dates of the competition overlap with another competition and not have the registration data mixed.</li> </ul> </li> </ul>
Functional Requirements – LEVEL 3	FR_07	<ul style="list-style-type: none"> <li>• Optimize database queries for music uploading:               <ul style="list-style-type: none"> <li>• When the festival chairman for a competition needs to upload music, they must wait for a considerable amount of time because there is 1183 songs that need to be uploaded to the central music database. Currently, ARIA will process all 1183 songs from a CSV file and will perform 1183 SQL queries. This is a brute-force approach, but modifying the SQL query code would require modification to another plugin, called GravityForms, that ARIA relies on.</li> </ul> </li> </ul>
	FR_08	<ul style="list-style-type: none"> <li>• Interact with additional APIs to enhance features:               <ul style="list-style-type: none"> <li>◦ There are several areas in the plugin that could presumably be enhanced by integrating third-party APIs. For example, the Google Maps API could be used in conjunction with the competition location so that students/teachers could have a visual representation of where a competition is taking place.</li> </ul> </li> </ul>

Table 1: Summary of functional and non-functional requirements. Level 1 Priority requirements correspond to the requirements which the team will complete by April 2016. Level 2 Priority requirements are requirements which the team will attempt to implement if time permits. Level 3 requirements will likely not be implemented by the end of the Spring 2016 semester, but would be a good advancement for the project.

### **2.3 USE CASE MODELING**

The goals of ARIA can be modeled with a series of use cases.

### 2.3.1 Use Case Diagram

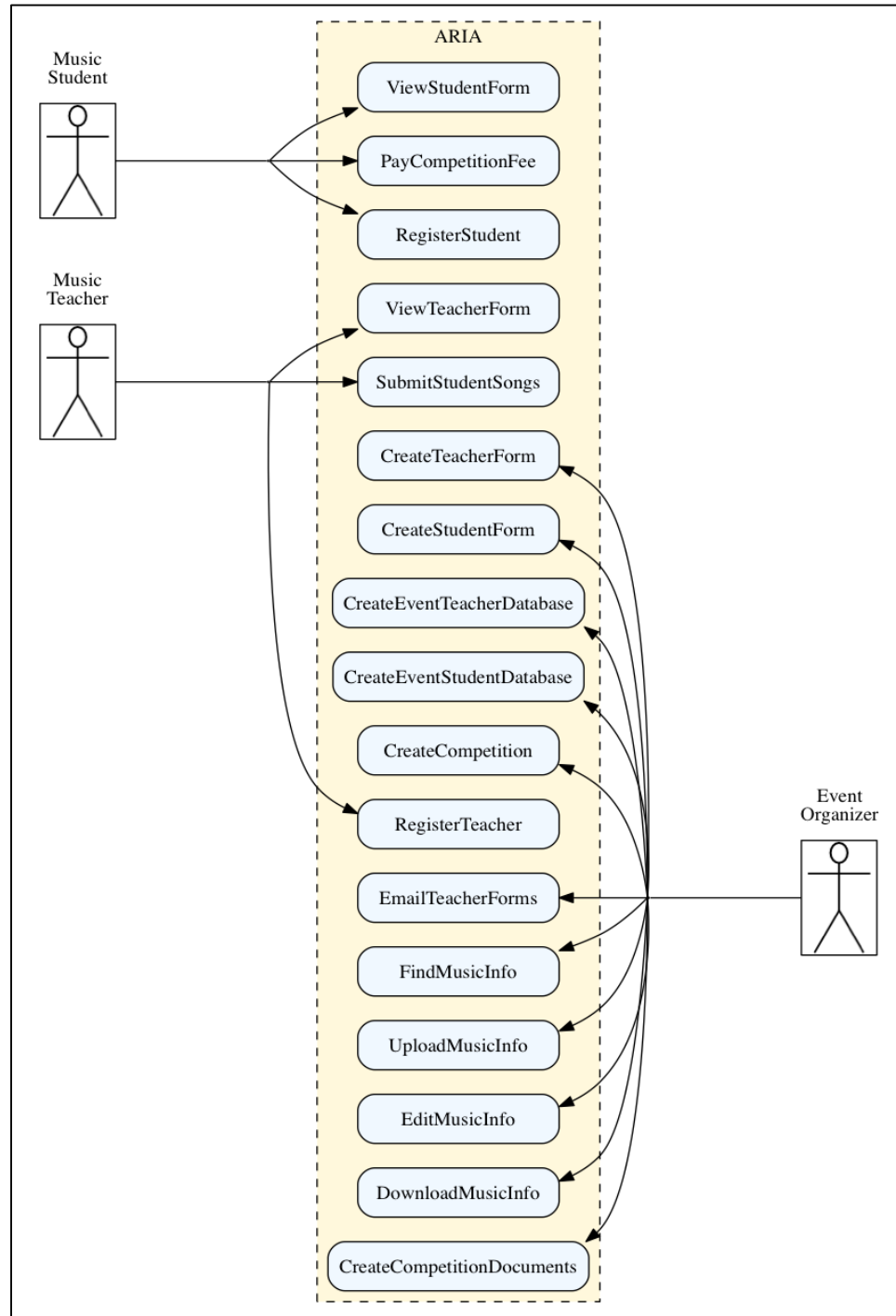


Figure 1: Diagram of project use cases. Scenario includes three actors and associated use cases.

### 2.3.2 Advanced Use Case Diagrams

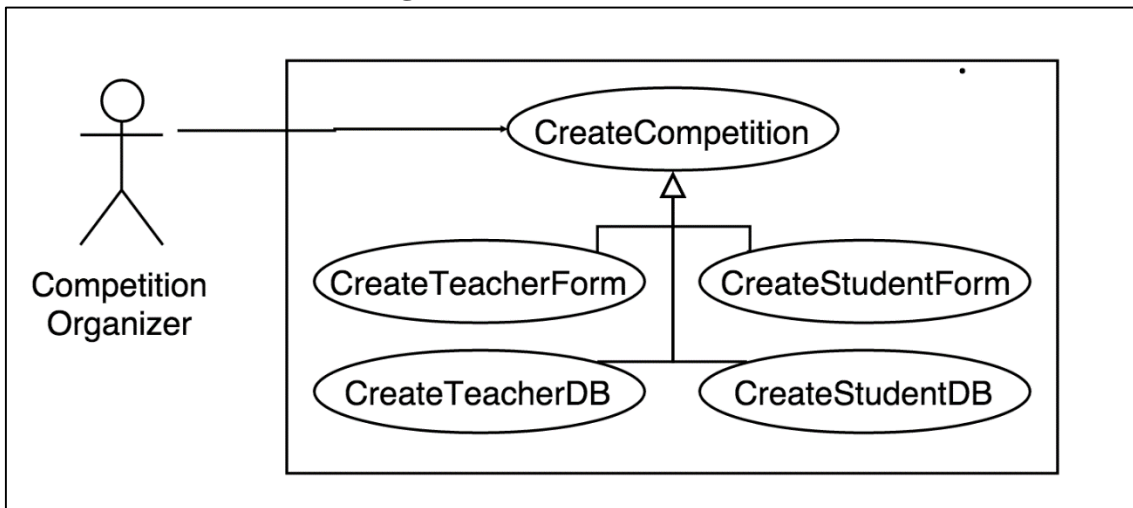


Figure 2: Advanced Use Case 1. Competition Creation Use Cases.

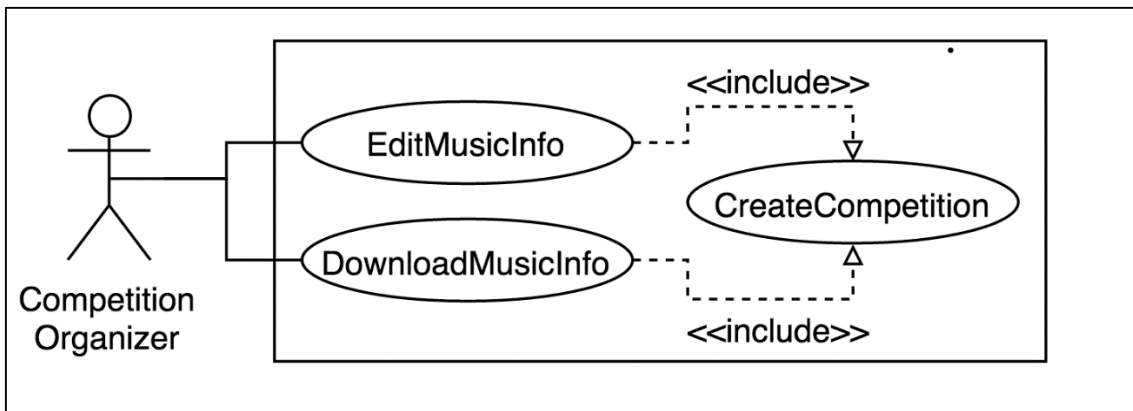


Figure 3 Advanced Use Case 2. Music Info inclusion.

### 2.3.3 Use Case Descriptions

Use cases are described in more detail below. Use cases written in italics are detailed in the Detailed Use Case section.

- ViewStudentForm (UC\_01)
  - Students will be able to view the competition form that will be used to register them for a specific competition.

- *PayCompetitionFee* (UC\_02)
  - Students will be able to pay for the competitions that they register for. The payment amount will differ based on the level of competition.
- *RegisterStudent* (UC\_03)
  - Students will have the opportunity to register for multiple competitions. Each competition has a different registration form so students can register for as many active competitions as they wish.
- *ViewTeacherForm* (UC\_04)
  - Music teachers will be able to view the competition form which will be used to register them for a specific competition and will contain information about their students.
- *SubmitStudentSongs* (UC\_05)
  - Music teachers will be able to select the songs which their students will be performing for a specific competition. This involves selecting the song, composer, and period of a given performance piece.
- *CreateTeacherForm* (UC\_06)
  - The event organizer will be able to create the form which will be submitted by teachers to register for a specific competition. This form will contain specific fields corresponding to the teacher's information.
- *CreateStudentForm* (UC\_07)
  - The event organizer will be able to create the form which will be published to the website for students to register for a specific competition. This form will contain specific fields corresponding to the student's information.

- CreateEventTeacherDatabase (UC\_08)
  - The event organizer will be able to create the database which will contain all of the information corresponding to a specific teacher, such as their name or email, which will be populated upon form submission.
- CreateEventStudentDatabase (UC\_09)
  - The event organizer will be able to create the database which will contain all of the information corresponding to a specific student, such as their name or email, which will be populated upon form submission.
- *CreateCompetition* (UC\_10)
  - The festival chairman will be able to dynamically create a competition by filling in the necessary details, such as competition dates and configuration options.
- RegisterTeacher (UC\_11)
  - The teacher will be able to register for a competition by filling out personal information, as well as specify song selection information for his or her students.
- EmailTeacherForms (UC\_12)
  - As soon as the students register for the competition, it will email his or her music teacher letting them know that the student has completed his or her part of the registration process
- FindMusicInfo (UC\_13)
  - The event organizer will be able to search through their uploaded music information in order to find specific music pieces.

- *UploadMusicInfo* (UC\_14)
  - The event organizer will be able to upload a file with all of the info about possible song selections for a specific music competition. This will be in the form of a CSV file.
- *EditMusicInfo* (UC\_15)
  - In collaboration with UC\_15, the festival chairman will have the ability to download the NNMTA CSV music file. The festival chairman can then edit this CSV file and re-upload the file. This will effectively update all of the NNMTA music.
- *DownloadMusicInfo* (UC\_16)
  - The festival chairman will have the ability to download a CSV file that contains all of the NNMTA music.
- *CreateCompetitionDocuments* (UC\_17)
  - The festival chairman will have the ability to generate all of the competition documents for a specific music competition. These can be judging forms, student schedule forms, and volunteer forms.

#### **2.3.4 Detailed Use Cases**

Select use cases from the comprehensive list above have been described in more detail, including relevant actors, preconditions, postconditions, and event flow.

<b>Use Case: ViewTeacherForm</b>	
Use Case ID	UC_04
Actor	Teacher
Precondition(s)	Student has registered for the event.
Flow of Events	<ol style="list-style-type: none"> <li>1. The use case starts when an actor is emailed a link in order to specify student song information.</li> <li>2. The page will allow the form to render if the link that was provided is correct.</li> <li>3. The page is prepopulated with the student's information.</li> <li>4. If the actor has registered a student beforehand the form is prepopulated with their user information.</li> <li>5. Else the fields will not be populated and the form will be rendered.</li> </ol>
Postcondition(s)	The form for teacher registration will be rendered to the screen.

Table 2: View Teacher Form use case details.

<b>Use Case: UploadMusicInfo</b>	
Use Case ID	UC_14
Actor	Organizer
Precondition(s)	<ol style="list-style-type: none"> <li>1. The Organizer is logged into wordpress.</li> </ol>
Flow of Events	<ol style="list-style-type: none"> <li>1. The actor will be prompted to select a csv containing music information.</li> <li>2. The actor will specify a csv to upload.</li> <li>3. The actor will be presented with a status bar presenting how many songs have been uploaded.</li> <li>4. The actor will be presented with confirmation that the music is uploaded.</li> </ol>
Postcondition(s)	<ol style="list-style-type: none"> <li>1. The information about music will be saved onto the database.</li> </ol>

Table 3: Upload Music Info use case details.

<b>Use Case: PayCompetitionFee</b>	
Use Case ID	UC_02
Actor	Students
Precondition(s)	1. Student Form filled out and validated.
Flow of Events	<ol style="list-style-type: none"> <li>1. The actor will input the type of their credit card.</li> <li>2. The actor will input the credit card number and expiration date.</li> <li>3. The actor will input their billing address.</li> <li>4. The actor will submit the request for payment.</li> <li>5. The actor will be notified of the status of the payment.</li> </ol>
Postcondition(s)	<ol style="list-style-type: none"> <li>1. The registration will be paid for.</li> <li>2. Money will be transferred from the Actor to the NNMTA.</li> </ol>

Table 4: Pay Competition Fee use case details.

<b>Use Case: CreateCompetition</b>	
Use Case ID	UC_10
Actor	Competition Organizer
Precondition(s)	1. Logged in
Flow of Events	<ol style="list-style-type: none"> <li>1. The competition organizer will specify the name of the event.</li> <li>2. The competition organizer will specify the date of the event.</li> <li>3. The competition organizer will specify any divisions of the competition.</li> <li>4. The competition organizer will specify the available rooms for the competition.</li> <li>5. The competition organizer will specify the approximate time for each performance.</li> <li>6. The competition organizer will denote the description of the event.</li> <li>7. The competition organizer will submit the competition.</li> </ol>
Postcondition(s)	<ol style="list-style-type: none"> <li>1. The student and teacher forms will be created for the event.</li> <li>2. Student and teacher databases will be created for the event.</li> </ol>

Table 5: Create Competition use case details.

## 2.4 REQUIREMENT TRACEABILITY MATRIX

	Represents a function requirement that <i>may</i> be needed for use case.
	Represents a function requirement that <i>must</i> be needed for use case.

	FR_01	FR_02	FR_03	FR_04	FR_05	FR_06	FR_07
UC_01							
UC_02							
UC_03							
UC_04							
UC_05							
UC_06							
UC_07							
UC_08							
UC_09							
UC_10							
UC_11							
UC_12							
UC_13							
UC_14							
UC_15							
UC_16							
UC_17							

Table 6: Requirements Traceability Matrix. This maps use cases to their applicable functional requirements.

### 3.1 ANALYSIS DIAGRAMS

ARIA can be broken into a series of analysis packages and classes.

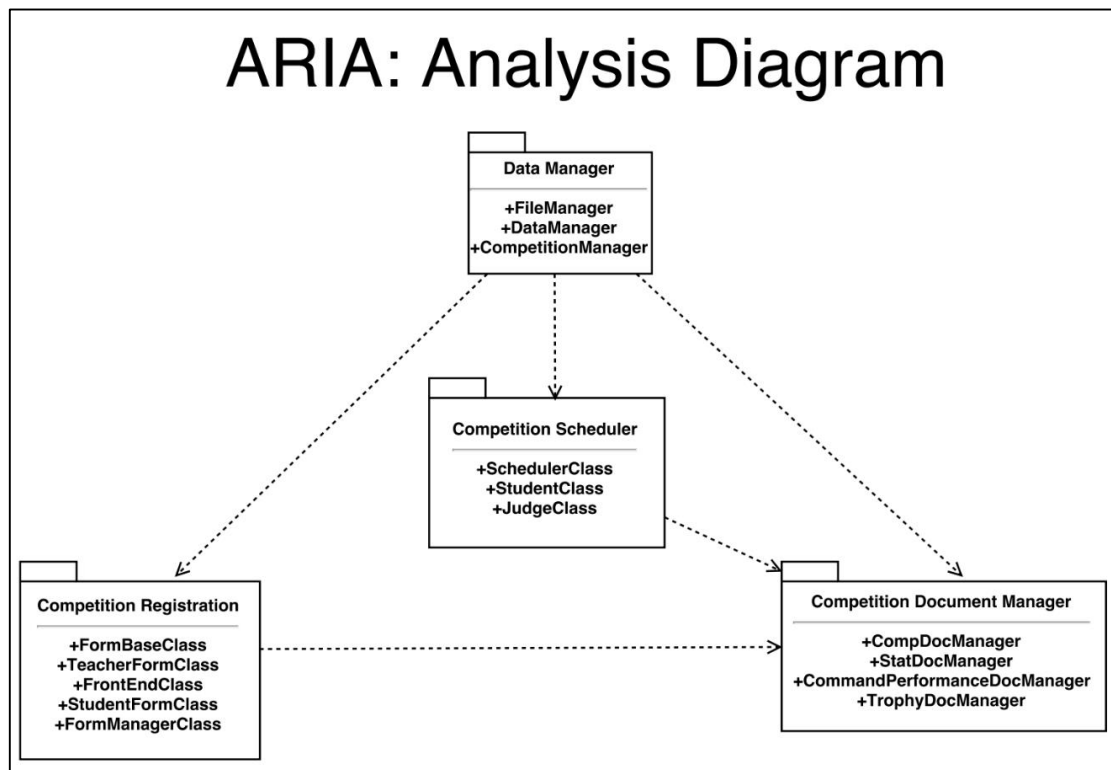


Figure 4: The diagram above represents the analysis model for ARIA. The diagram is composed of a data manager module, a competition registration module, a competition scheduler module, and a competition document manager module. These packages not only depict the high-level structure of ARIA, but also pertain to the fundamental functional requirements that are defined earlier in this document.

### 3.2 Analysis Packages

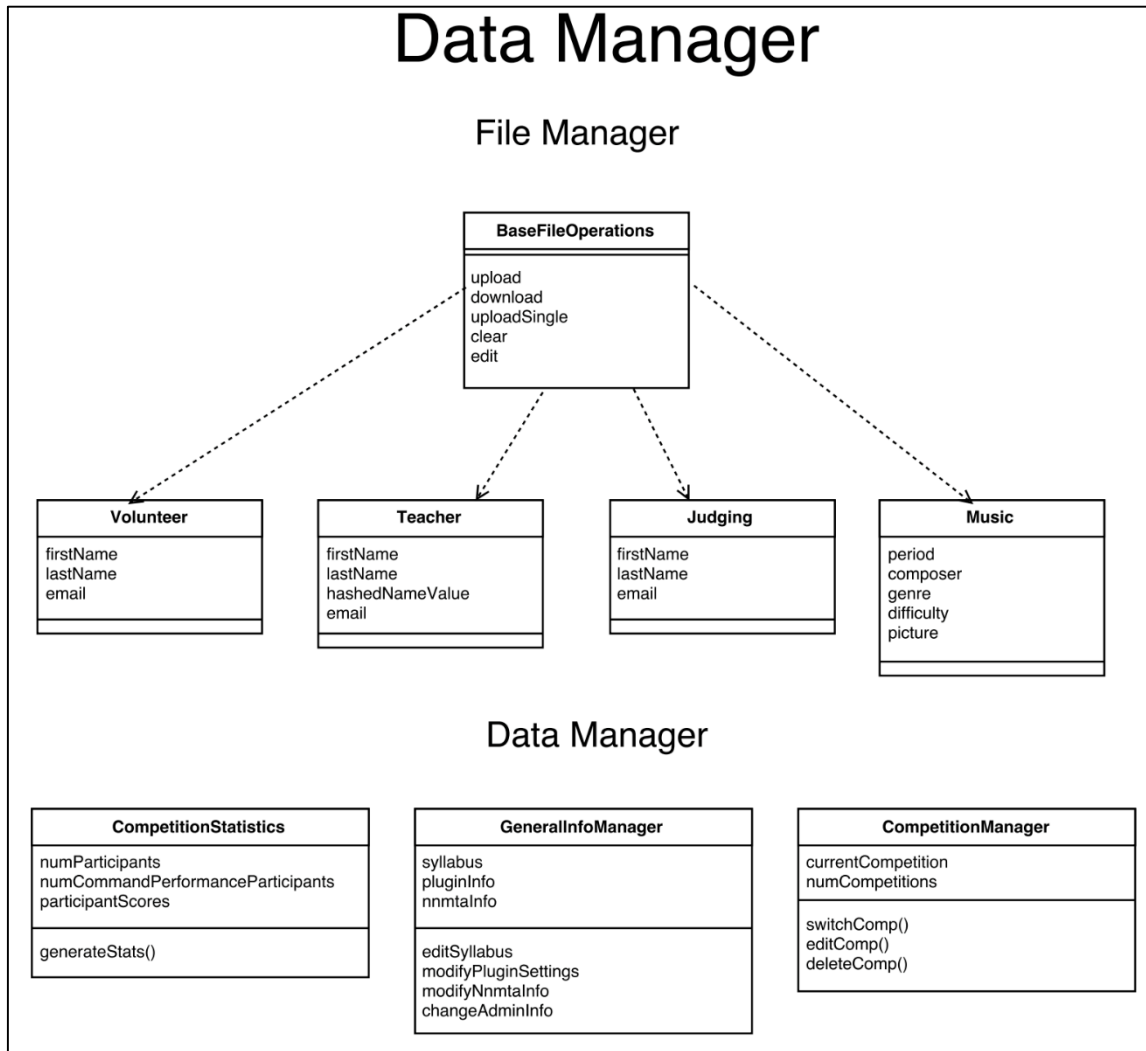


Figure 5: The diagram above represents the data manager analysis package. Its primary responsibility is the handling of ARIA's data, whether it corresponds to data accumulated from files or data that exists are part of the plugin. The file manager is further broken down into a management system for volunteers, teachers, judges, and music, all of which are input from files. The data manager, as previously mentioned, handles all other data associated with the ARIA plugin.

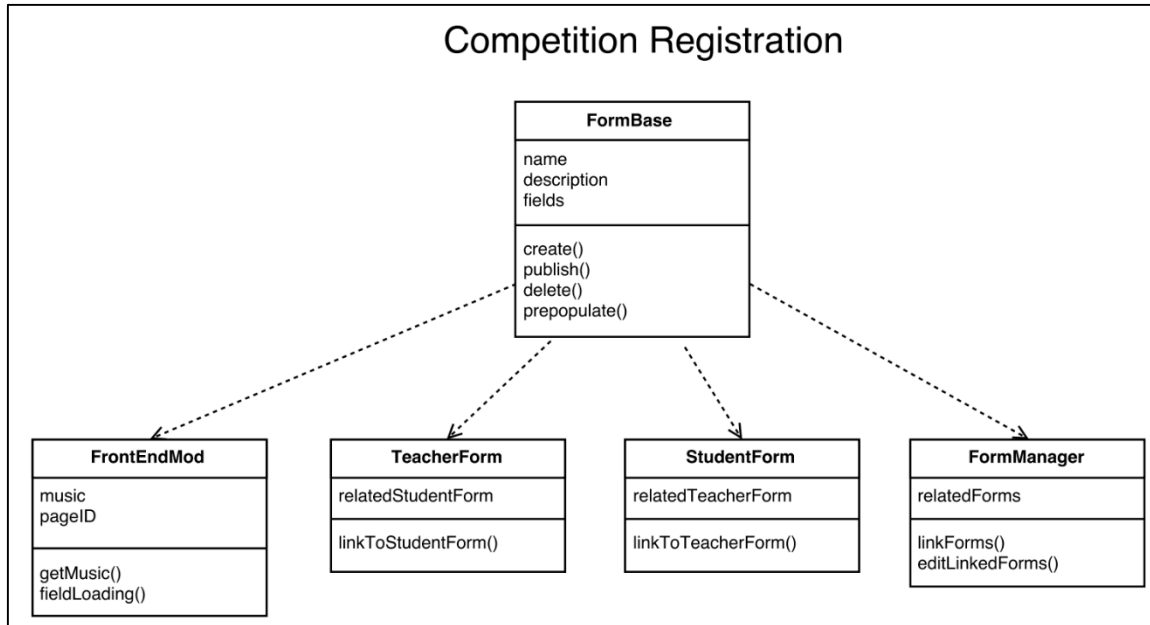


Figure 6: The diagram above represents the competition registration analysis package. The intended purpose of this analysis package is the management of all forms associated with ARIA. This includes the teacher and student forms, how these forms are displayed on the front end of the plugin (FrontEndMod) and also a backend interface for the festival chairman (FormManager) that can be used to manage any form. Each of these derives information from a base form module.

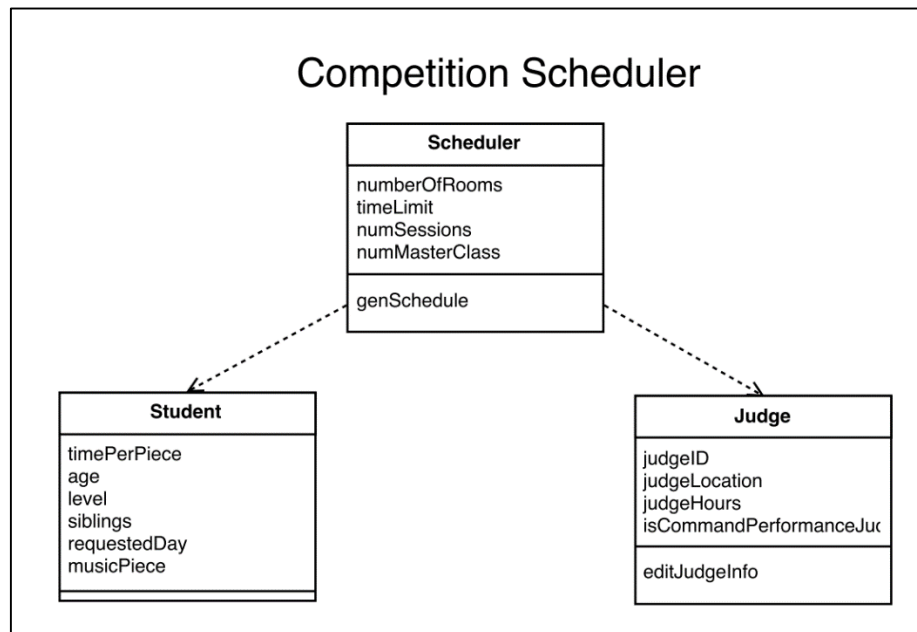


Figure 7: The diagram above represents the competition scheduler package. The modules included are scheduler, student, and judge. These modules are created so that the scheduler

is able to generate the schedule for a competition which meets the requirements for both students and judges without any conflicts.

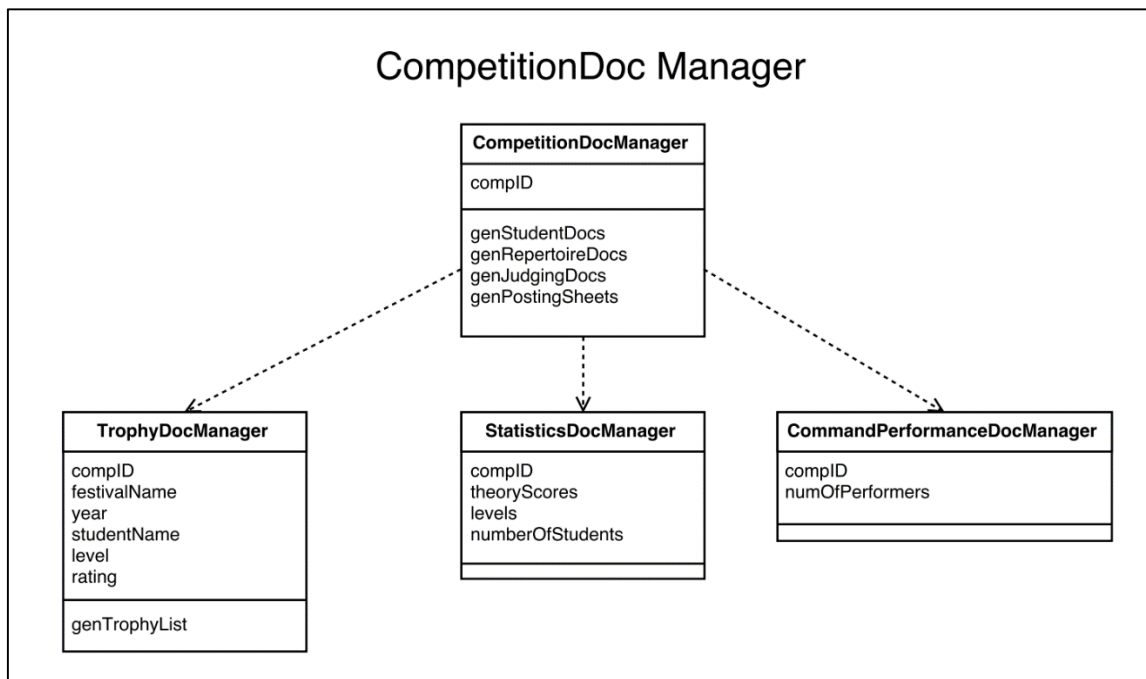


Figure 8: The diagram above represents the competition document manager package. The modules consists of a competition document manager, trophy document manager, statistic document manager, and a command performance document manager. The main functionality for each module is to hold the data needed to generate documents for the competition once the time comes.

## **4.1 DESIGN MODEL**

### **4.1.1 Architectural Design**

- **Structural Diagram**

- Figure 9 represents a high-level structural diagram of ARIA. The user interacts with ARIA through the GUI subsystem, which provides a gateway to the functionality of the other subsystems. Specifically, the GUI subsystem interacts with the competition, registration, music, and document subsystems. Continuing, the registration subsystem interacts with the scheduler subsystem and the document subsystem communicates with the statistics subsystem. All subsystems that are considered as “ARIA processing” (in the diagram) require the functionality provided by the Gravity Forms plugin and the GUI subsystem needs the WordPress code to operate as intended.

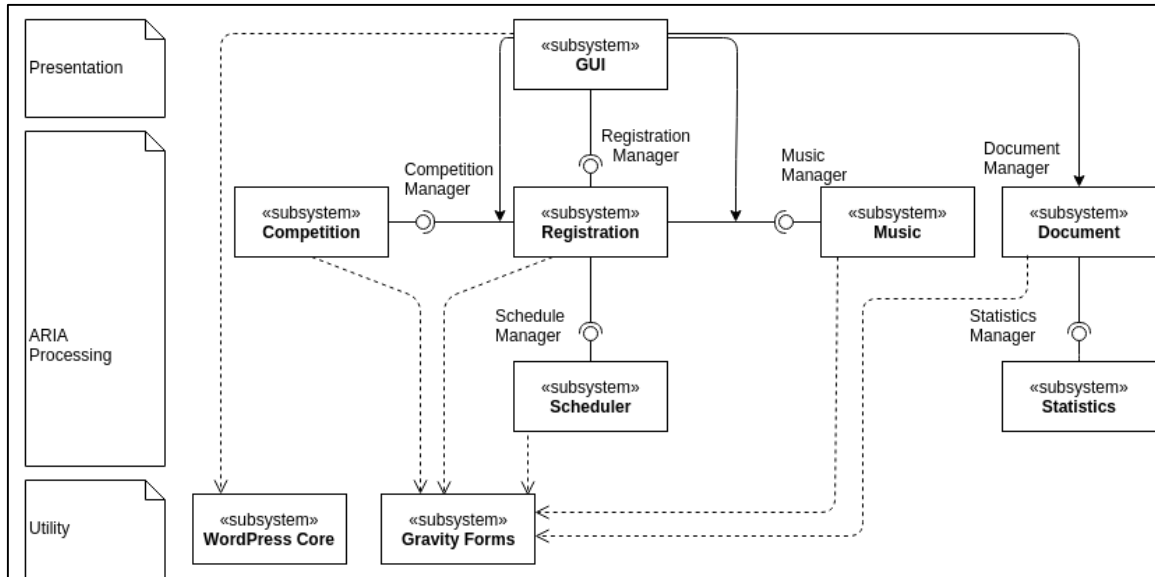


Figure 9: High-level structural diagram of ARIA.

- **Activity Flow**

- Figure 10 depicts the high-level flow of activity of ARIA. Upon activation of the plugin, the chairman is able to select different actions: uploading music, creating a competition, scheduling a competition, or generating documents. The activity diagrams for the competition creation, student registration, teacher registration, and document generation are depicted in the Detailed Design section. The festival chairman has the option to repeat any of the main functions of ARIA or to deactivate the plugin.

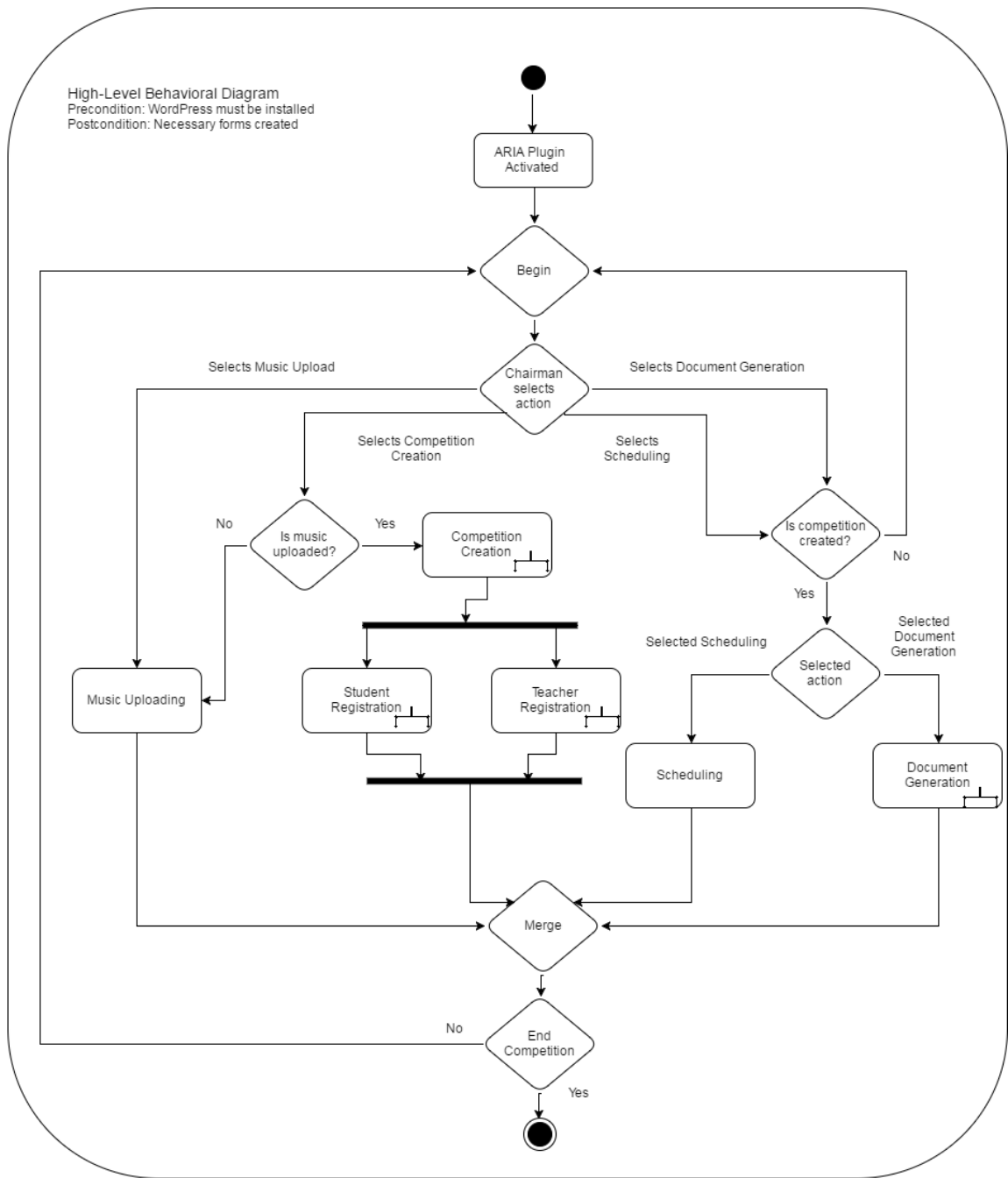


Figure 10: High-level activity flow.

## 4.2 CLASS DIAGRAMS

- GUI Design Class
  - The GUI design class in Figure 11 relies on “Activator” and “Deactivator” objects in order to correctly perform initialization and deinitialization correctly. The “ARIA” object, which makes use of both the “Activator” and “Deactivator” object, also requires a “Loader” object. The “Loader” object is responsible for maintaining all of the actions and filters that are attached to WordPress core. The “ARIA” object uses these objects in order to correctly generate the GUI that the admin sees in the WordPress dashboard.

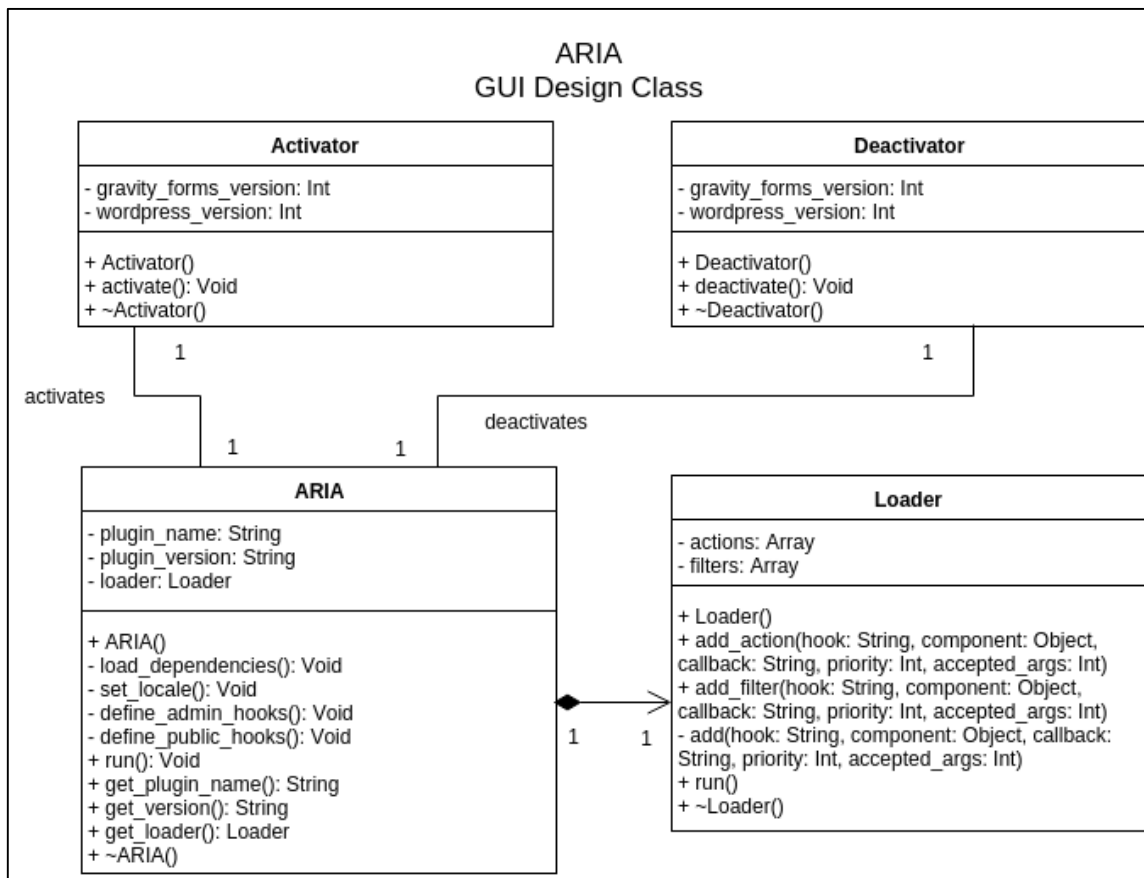


Figure 11: GUI design class.

- Scheduler Design Class
  - The scheduler design class depicted in Figure 12 consists of a main “Scheduler” object. The scheduler object is responsible for taking a student as input and scheduling the student. Inside of the scheduler object is a multidimensional array of “TimeBlock” objects. The “Scheduler” object passes the student to be scheduled into the appropriate “TimeBlock” object, which then passes the student to be scheduled to one of its internal “Section” objects (each “TimeBlock” object has an array of “Section” objects). Each “Section” object maintains an array of students. The student to be scheduled will be added to the “Section” objects array of students. Finally, each “Student” object holds an array of “Song” objects.

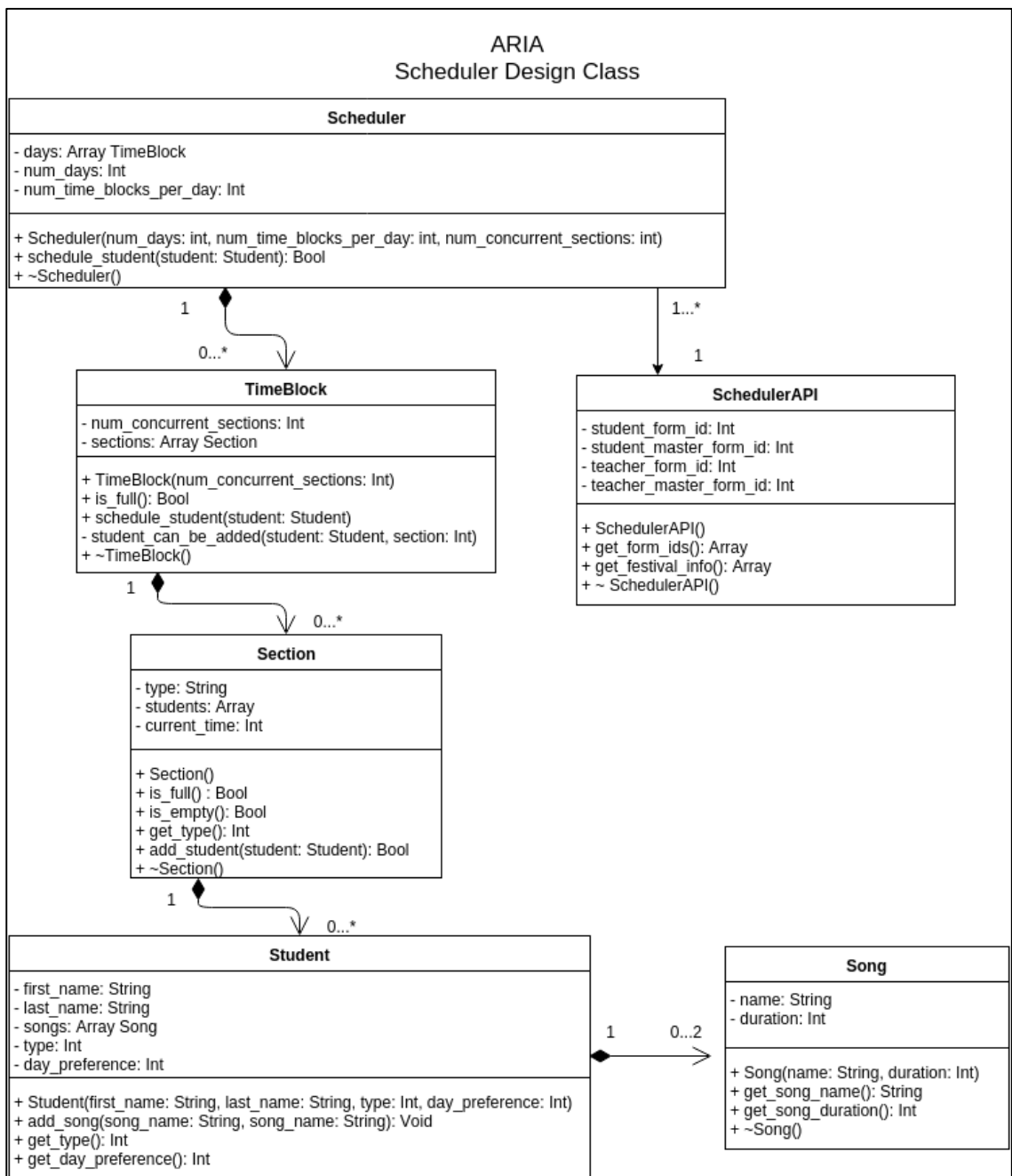


Figure 12: Scheduler design class

- Competition Design Class
  - The main object in subsystem is the “CreateCompetition” object, whose purpose is to house all of the functionality required for creating

competitions. To achieve this functionality, the “CreateCompetition” object communicates directly with a “CreateMasterForms” objects (creates backend databases needs for competition registration), a “TeacherUpload” object (uploads a list of teachers to the current competition), and a “CompetitionAPI” object (maintains a communal list of functions used by all objects in this subsystem).

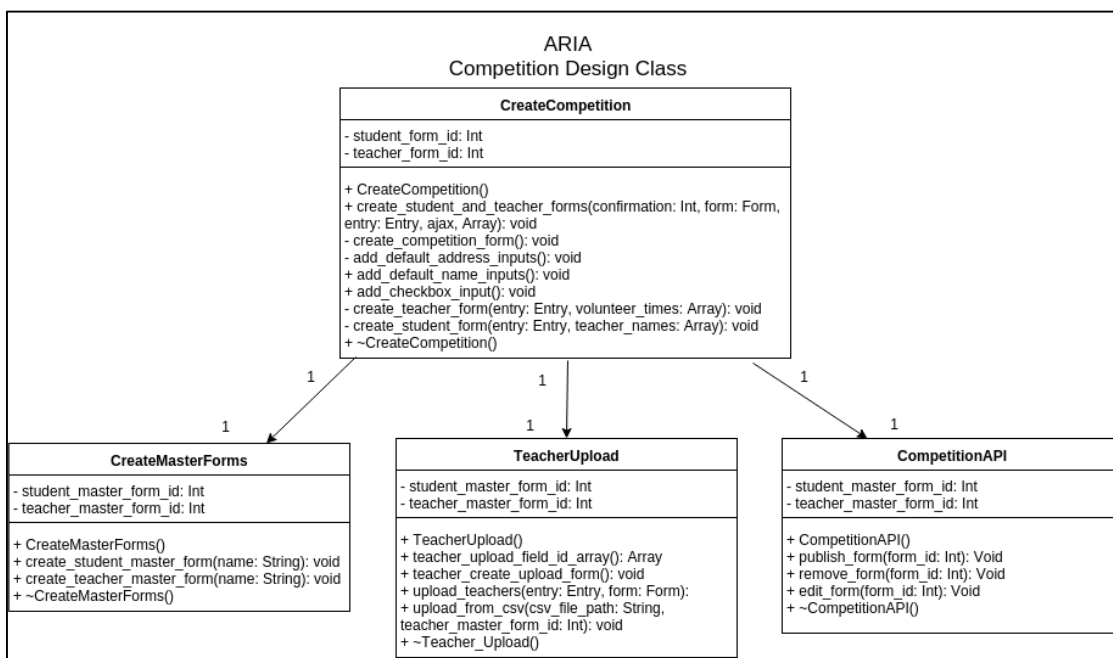


Figure 13: Competition design class.

- Registration Design Class
  - Shown in Figure 14, the “RegistrationFormHooks” object defines the functionality that is required when students and teachers sign up for an NNMTA music competition. To achieve this functionality, each “RegistrationFormHooks” object communicates with a “RegistrationHandler” object (provides ancillary functionality to the

student/teacher sign-up process) and a “RegistrationAPI” object (provides a communal list of functions that are used throughout ARIA in regards to registration).

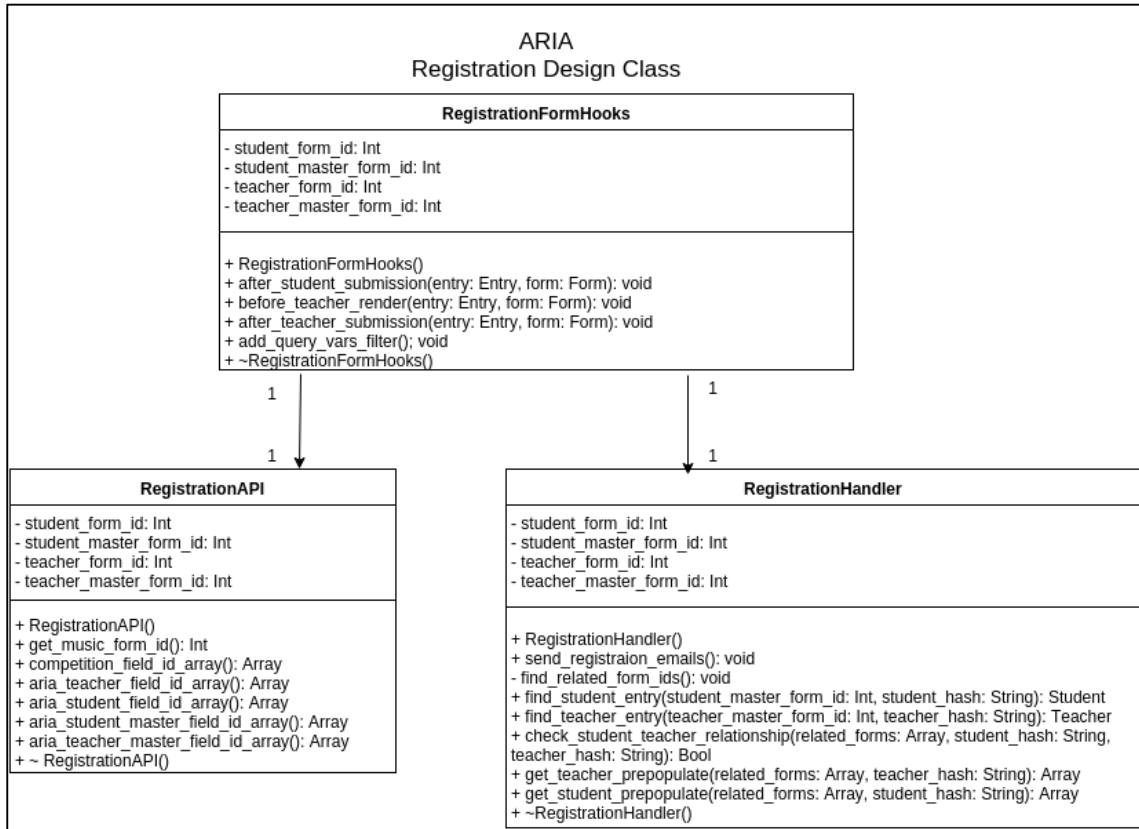


Figure 14: Registration design class.

- Music Design Class
  - The “MusicManager” object in Figure 15 defines all of the functionality that is needed to upload and modify the music that belongs to the NNMTA.

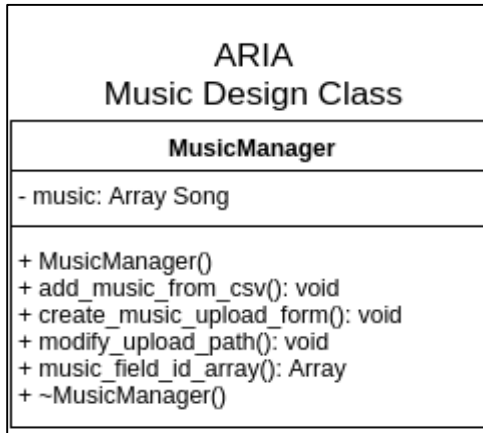


Figure 15: Music design class.

- Document Design Class
  - In Figure 16, the “DocumentManager” object defines all of the functionality that is needed to print all of the documents required by the NNTMA on music competition days.

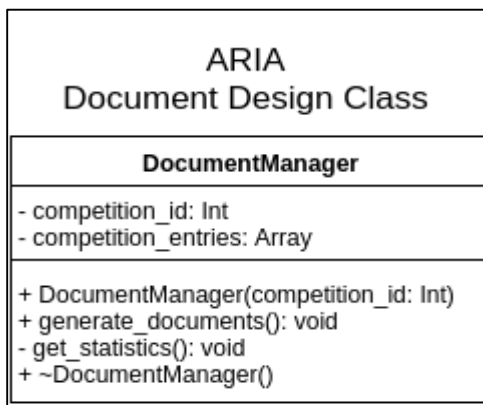


Figure 16: Document design class.

- Statistics Design Class
  - The “StatisticsManager” objects in Figure 17 defines all of the functionality that is needed to generate all of the statistics for the NNMTA.

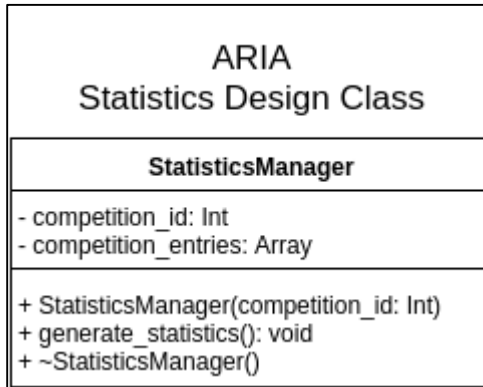


Figure 17: Statistics design class.

### 4.3 PROGRAM UNITS

- Create\_competition\_activation
  - This function will create the form that can create new music competitions. This function is called in "class-aria-activator.php" and is responsible for creating the form that allows the festival chairman to create new music competitions (if this form does not already exist). If no such form exists, this function will create a new form designed specifically for creating new music competitions.
  - Precondition: none
  - Postcondition: A new form for creating new music competitions will be created.
- Create\_teacher\_and\_student\_forms
  - This function will create new registration forms for students and parents. This function is responsible for creating new registration forms for both students and parents. This function will only create new registration forms for students and parents if it is used ONLY in conjunction with the form used to create new music competitions.

- Precondition: entry in create competition submitted.
  - Postcondition: student and teacher forms for a competition is made.
- Create\_competition\_form
  - This function will create a new form for creating music competitions. This function is responsible for creating and adding all of the associated fields that are necessary for the festival chairman to create new music competitions.
  - Precondition: none
  - Postcondition: The forms necessary will be created for a competition.
- Add\_default\_address\_inputs
  - This function is responsible for adding some default address field values. This function is used to pre-populate the address fields of a gravity form with some generic, default values.
  - Precondition: field initialized
  - Postcondition: inputs will be added to an address field
- Add\_default\_name\_inputs
  - This function is responsible for adding default name inputs to a name field.
  - Precondition: field initialized
  - Postcondition: inputs will be added to a name field
- Add\_checkbox\_input
  - This function is responsible for adding checkbox inputs to a checkbox field.
  - Precondition: field initialized
  - Postcondition: inputs will be added to a checkbox field

- Aria\_create\_teacher\_form
  - This function will create a new form for the teachers to use to register student information. This function is responsible for creating and adding all of the associated fields that are necessary for music teachers to enter data about their students that are competing.
  - Precondition: a competition is being created.
  - Postcondition: a form for teacher registration is created.
- Create\_student\_form
  - This function will create a new form for student registration. This function is responsible for creating and adding all of the associated fields that are necessary for students to enter data about their upcoming music competition.
  - Precondition: a competition is being created.
  - Postcondition: a form for student registration is created.
- After\_student\_submission
  - This function will be the hook that is called after a student submits their information for a new music competition. This function will take all of the information that the student submitted and update corresponding data in the student form, the student master form, and the teacher master form.
  - Precondition: a student has submitted their registration.
  - Postcondition: an entry in teacher database and student database will be created and an email to their piano teacher will be sent.
- Before\_teacher\_render

- This function will be the hook that is called when navigating to the teacher registration page. The function will either let the user see the form or navigate to the home page if the link used to navigate to the page was correct or incorrect.
- Precondition: none
- Postcondition: the form is rendered or the user is navigated to the homepage.
- `After_teacher_submission`
  - This function will be the hook that is called after a teacher submits information for a particular student. This function will take all of the information that the teacher submitted and update corresponding data in the teacher form, the student master form, and the teacher master form.
  - Precondition: the teacher is submitting a form.
  - Postcondition: the corresponding entries in the teacher database and student database will be created.
- `Add_query_vars_filter`
  - This function will expose the new, custom query variables to `WP_Query`. In order for ARIA's query hash method to work, specific query vars (the query vars that will be added to URLs) need to be added to the public query variables that are available to `WP_Query`. This function is responsible for adding these query vars to the `$query_vars` property of `WP_Query`.
  - Precondition: none
  - Postcondition: Wordpress will be able to detect values in the url.
- `Create_student_master_form`

- This function will create the form that will be the source of truth for a certain competitions students. This function is called in "class-aria-create-competition.php" and is responsible for creating the student master form. This form is the absolute source of truth for the students of any given competition. Entries in other forms will update entries in this form.
- Precondition: a competition is in the process of being created.
- Postcondition: A form will be created to hold student information.
- Create\_teacher\_master\_form
  - This function will create the form that will be the source of truth for a certain competitions teachers This function is called in "class-aria-create-competition.php" and is responsible for creating the teacher master form. This form is the absolute source of truth for the teachers of any given competition. Entries in other forms will update entries in this form.
  - Precondition: a competition is in the process of being created.
  - Postcondition: A form will be created to hold student information.
- Add\_music\_from\_csv
  - This function will parse the contents of the csv file and upload content to the NNMTA music database. Using the csv file that the user has uploaded, this function will parse through the music content for each song and add it to the NNMTA music database.
  - Precondition: none
  - Postcondition: the database will be filled with new music entries.
- Create\_music\_upload\_form

- This function is responsible for creating the NNMTA music uploading form if it does not exist. This function is intended to be used in the event where the form for uploading music does not previously exist. If no such form exists, this function will create the form used for uploading music.
- Precondition: none
- Postcondition: the form for adding music to a the database is created.
- Create\_nnmta\_music\_form
  - This function is responsible for creating the NNMTA music form if it does not previously exist. This function is intended to be used in the event where the festival chairman tries to upload music to the NNMTA database but no such form exists for adding music.
  - Precondition: none
  - Postcondition: The NNMTA music form will be created.
- Remove\_all\_music\_from\_nnmta\_database
  - This function will remove all of the music from the NNMTA music database. This function was created to support the scenario when the festival chairman needs to update the music in the NNMTA music database. In order to do this, all of the existing data is removed from the database prior to adding all of the new data. This ensures that the new data is added appropriately without accidentally adding old, possibly unwanted music data.
  - Precondition: none
  - Postcondition: the music database will be cleared.

- `Modify_upload_path`
  - This function will change the default file path for uploaded files. In order to upload music from a file, we need to know where the music file resides. This function will set a pre-determined file path so the music data can be read from.
  - Precondition: none
  - Postcondition: file uploads will upload to a specified path.
- `Get_create_competition_form_id`
  - This function will find the ID of the form used to create music competitions. This function will iterate through all of the active form objects and return the ID of the form that is used to create music competitions. If no music competition exists, the function will return -1.
  - Precondition: none
  - Postcondition: none
- `Get_teacher_upload_form_id`
  - This function will find the ID of the form used to upload music teachers. This function will iterate through all of the active form objects and return the ID of the form that is used to upload music teachers. If no such form exists, the function will return -1.
  - Precondition: none
  - Postcondition: none
- `Get_song_upload_form_id`

- This function will find the ID of the form used to upload songs. This function will iterate through all of the active form objects and return the ID of the form that is used to upload music to the NNMTA music database.
- Precondition: none
- Postcondition: none
- `Get_nnmta_database_form_id`
  - This function will find the ID of the form used as the NNMTA music database. This function will iterate through all of the active form objects and return the ID of the form that is used to store all of the NNMTA music.
  - Precondition: none
  - Postcondition: none
- `Send_registration_emails`
  - This function is called after a student submits their registration and sends an email to their teacher giving them information on how to finalize the registration of their students.
  - Precondition: Student Submitting their registration
  - Postcondition: Teacher is sent an email.
- `Find_related_forms_ids`
  - This function will return an associative array that maps the titles of the associated forms in a music competition (student, student master, teacher, and teacher master) to their respective form IDs.
  - Precondition: none
  - Postcondition: none

- Find\_student\_entry
  - This function will search through the student-master form and check to see if a particular student exists. If a student exists within the student- master form of a particular competition, then the entry for that student will be returned. Otherwise, if no such student exists, the function will return false.
  - Precondition: none
  - Postcondition: none
- Find\_teacher\_entry
  - This function will search through the teacher-master form and check to see if a particular teacher exists. If a teacher exists within the teacher master form of a particular competition, then the entry for that teacher will be returned. Otherwise, if no such teacher exists, the function will return false.
  - Precondition: none
  - Postcondition: none
- Check\_student\_teacher\_relationship
  - This is a function that is used to check if a student is assigned to a teacher.
  - Precondition: none
  - Postcondition: none
- Get\_teacher\_pre\_populate
  - This is a function that is used to get the pre-population values for a specific teacher to be rendered on the teacher registration form.
  - Precondition: none
  - Postcondition: none

- Exception: the teacher doesn't exist.
- Get\_teacher\_pre\_populate
  - This is a function that is used to get the pre-population values for a specific student to be rendered on the teacher registration form.
  - Precondition: none
  - Postcondition: none
  - Exception: The student doesn't exist.
- CalculateSig
  - This is a front-end function as part of the teacher registration process used to calculate the signature for the URL used in GET requests sent by the forms.
  - Precondition: Crypto script is loaded.
  - Postcondition: none
- Get\_songs
  - This is a front-end function as part of the teacher registration process which retrieves all of song information specifically for the students level using a GET request to the Gravity Forms Web API.
  - Precondition: Student level is known, Gravity Forms is functioning.
  - Postcondition: Song information is loaded.
- Store\_periods
  - This is a front-end function as part of the teacher registration process which stores all of the possible time periods during the song selection process.

- Precondition: Song periods have been loaded into one of the song's dropdown.
  - Postcondition: Song periods are stored into an array for later access.
- Load\_composers
  - This is a front-end function as part of the teacher registration process which populates the composer dropdown menu with all composers from the selected period.
  - Precondition: Period is selected.
  - Postcondition: Composer dropdown is populated.
- Load\_periods
  - This is a front-end function as part of the teacher registration process which restores all song periods and is used for the mutual exclusion logic of period selection.
  - Precondition: none
  - Postcondition: All periods are restored to specified dropdown.
- Get\_Music\_Form\_ID
  - This is a front-end function as part of the teacher registration process which uses a GET request to retrieve the form ID number of the associated music database form for the current competition.
  - Precondition: Current competition ID is known.
  - Postcondition: Music form ID is known.
- toTitleCase

- This is a front-end function as part of the teacher registration process which reformats the given string into title case format for consistent presentation.
- Precondition: none
- Postcondition: none
- `Schedule_student`
  - This function will schedule a student depending on which day they had requested when they registered for a competition.
  - Precondition: the student has yet to be scheduled for the given competition.
  - Postcondition: the student has been scheduled according to their request and the student's parents are emailed with the student's schedule time.
- `Add_student`
  - If the current section matches the type of student competing (traditional, master-class, non-competitive, or command performance) and the current section is not full, then the incoming student object passed as a parameter will be added to the list of students competing in the current section.
  - Precondition: none
  - Postcondition: the student has been scheduled to a given time section for the competition.
- `Student_can_be_added`
  - This function is solely meant to simplify the condition that checks to see if a student can be added to a section in the function `schedule_student`.

- Precondition: none
- Postcondition: the student has been given permission or has been rejected from the given time section.

#### **4.4 DETAILED DESIGN**

Figure 18 - Figure 21 depict detailed activity flow for specific activities related to ARIA's functionality. These activities are listed as modules within the overall activity flow of ARIA. This includes competition creation, student registration, teacher registration, and document generation.

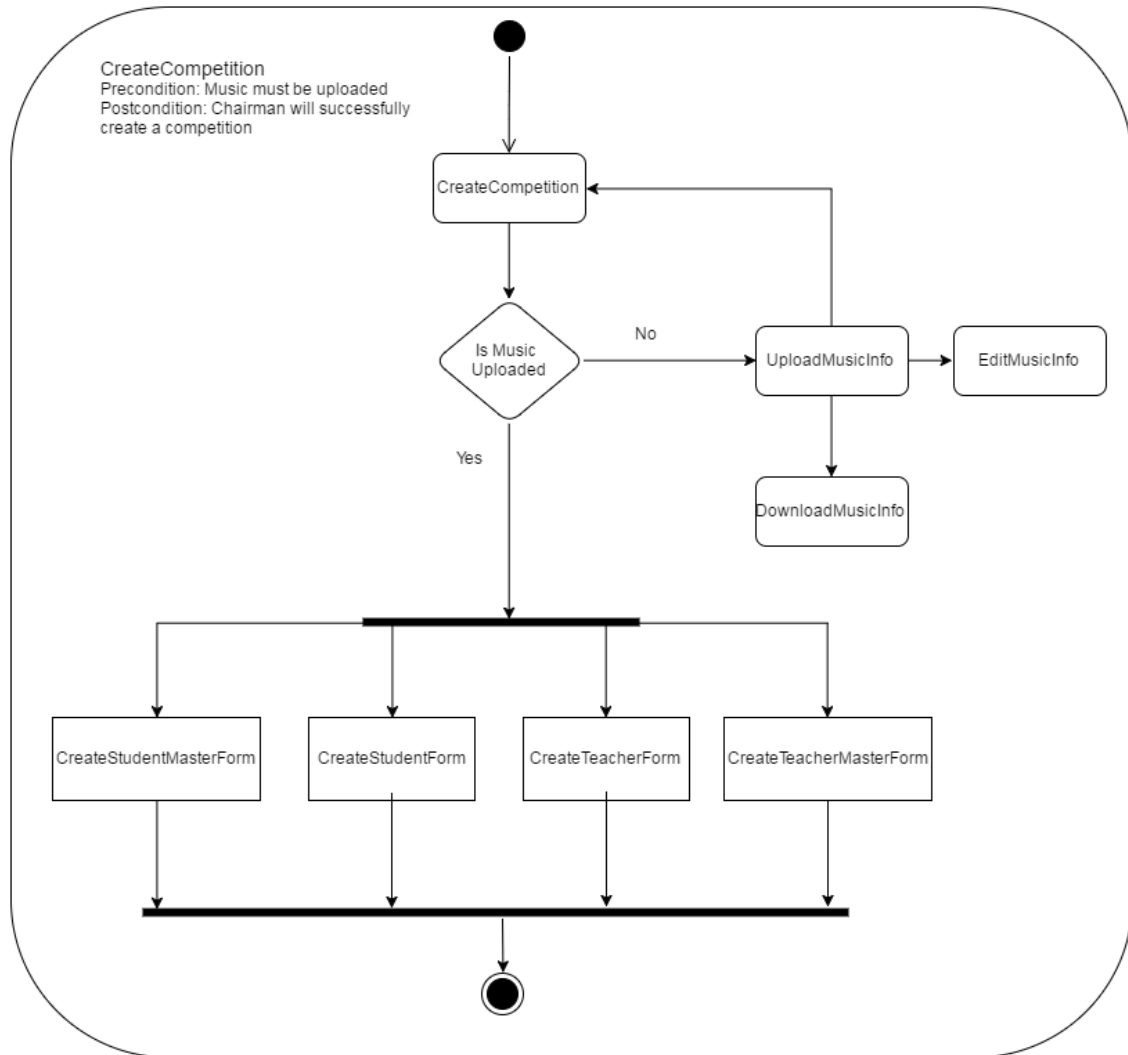


Figure 18: The activity diagram depicts the flow of activity for the process of creating a competition. If music data has not been uploaded, the chairman will be able to upload, edit, and download music. Once music data has been uploaded, the student and teacher forms are created along with the student and teacher master forms. Upon completion of these tasks, the activity terminates.

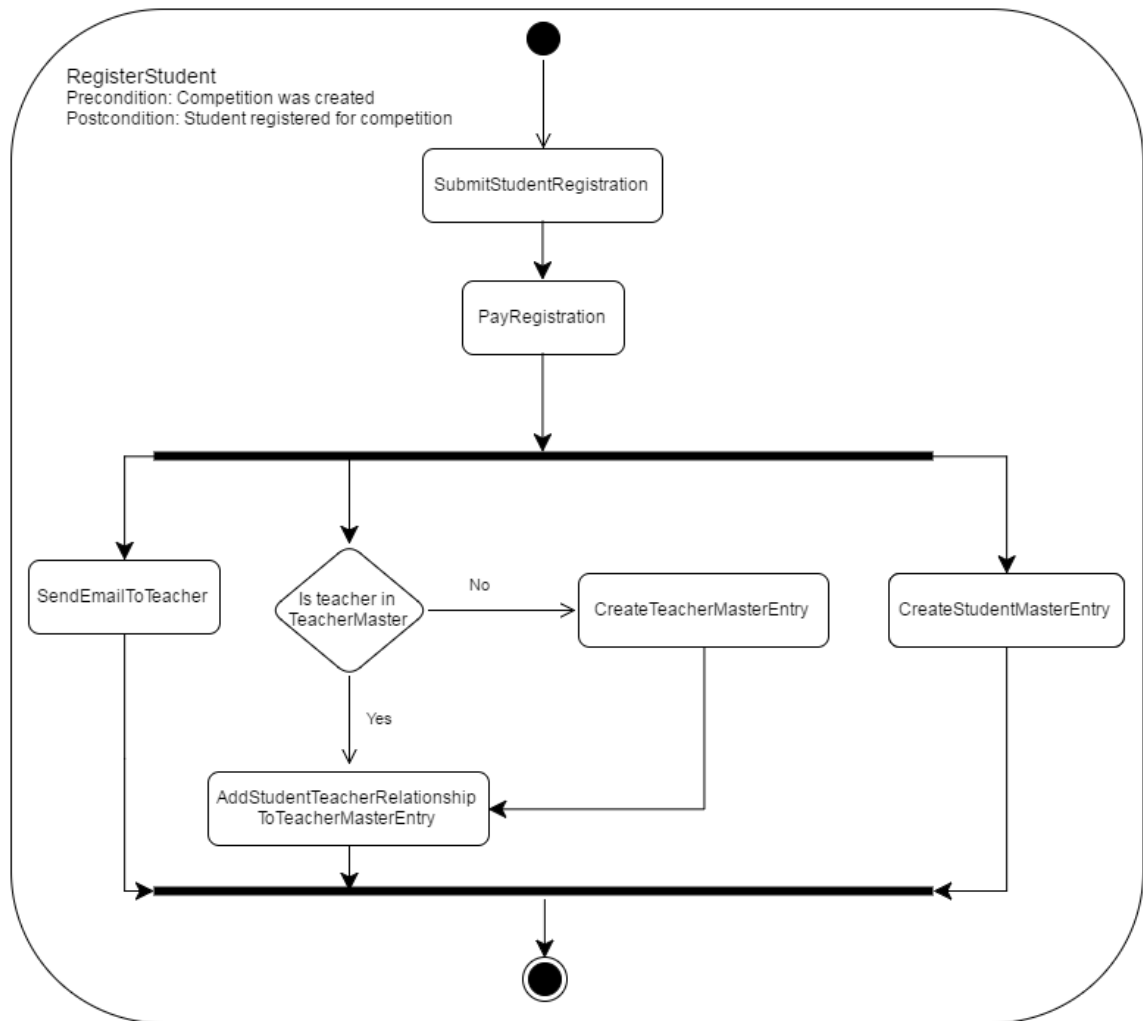


Figure 19: The above activity diagram depicts the student registration process. Once the student submits the registration form, they must pay for the competition. Upon completion of this, an email is sent to the teacher notifying them that one of their students has registered. Additionally, an entry in the teacher master is created if necessary, and the relationship is added into the teacher master entry. An entry in the student master form is also created simultaneously. Upon termination of all these flows, the activity terminates.

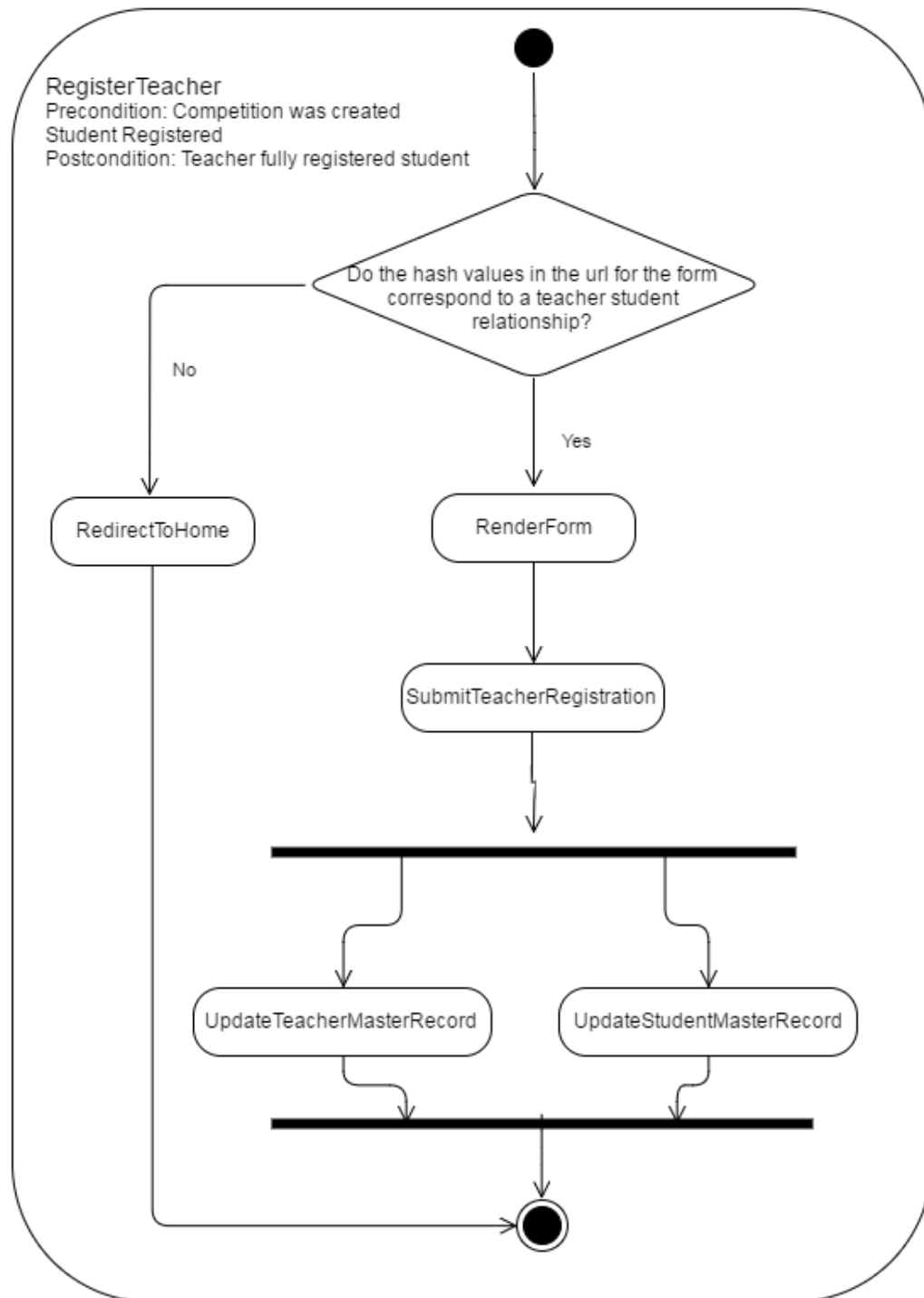


Figure 20: The above activity diagram depicts the teacher registration process. To begin, the hash values are validated. If the hash values are incorrect, then the user is redirected and activity terminates. Otherwise, the form is rendered with some pre-populated values. After the teacher submits the form, the entries in the teacher and student master database are updated.

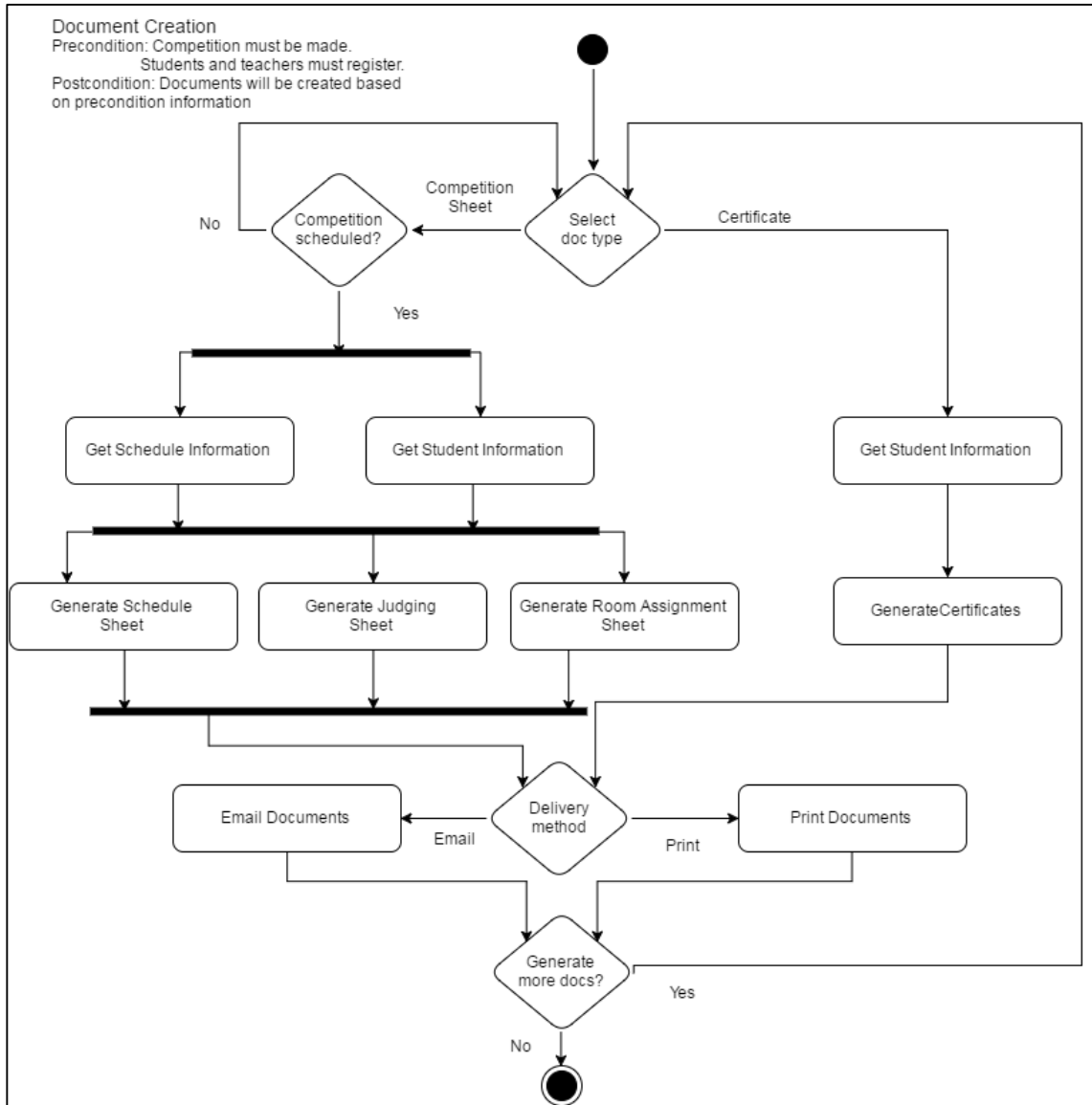


Figure 21: The activity diagram above depicts the process of document generation. The chairman is able to select which type of document to generate. If they select a competition sheet, then a competition must already have been scheduled. If so, the information for the competition schedule and students are fetched. With this information, necessary documents are generated. If the chairman selects a certificate type to be generated, then student information is fetched and certificates are generated. After document generation, the chairman selects whether to print or email the documents. After this is completed, the chairman is given the option to generate more documents. If they do not wish to generate more documents, activity terminates.

## 4.5 DATA DESIGN

The data stored for ARIA is organized with the following information.

- Music Data
  - All of the NNMTA music data is held in a centralized database and consists of the following data:
    - Name
    - Composer
    - Level
    - Period
    - Catalog number (defined by NNMTA).
- Competition Data
  - All of the competition data for an NNMTA music competition is held in a centralized database (per competition) and consists of the following data:
    - Name
    - Start date
    - End date
    - Location
    - Address
    - City
    - State
    - Zip
    - Country

- Student registration start date
  - Student registration end date
  - Teacher registration start date
  - Teacher registration end date
  - Volunteer times
  - Teacher Data (path to csv file)
  - Number of traditional sections
  - Number of masterclass sections
  - Number of non-competitive sections
  - Section length (minutes)
  - Number of judges per section
  - Number of command performances
  - Command performance start date
  - Command performance start time
  - Theory score needed for competition
- Student Data
    - All of the student data is held in a centralized database and consists of the following data:
      - Parent name
      - Parent email
      - Student name
      - Student birthday
      - Teacher name

- Day available
  - Preferred command performance time
  - First period
  - First composer
  - First selection
  - Second period
  - Second composer
  - Second selection
  - Theory score
  - Competition format
  - Timing of piece
  - Student hash value
  - Student level
- Teacher Data
    - All of the teacher data is held in a centralized database and consists of the following data:
      - Students (that belong to the given teacher)
      - Name
      - Email
      - Phone
      - Teacher hash value
      - Student hash values
      - Volunteer preference

- Volunteer time

#### 4.6 USER INTERFACE DESIGN

ARIA's main user interface deals with the creation and management of various forms and web pages. Important aspects of these are described in the illustrations below.

### ARIA: Create a Competition

Welcome! Please submit information for all of the fields in the form below in order to create a new NNMTA music competition.

**Competition Name**

**Competition Start Date**

**Competition End Date**

**Competition Location**

Street Address

Address Line 2


Illustration 1: Sample competition creation form. When ARIA is activated, a WordPress page with this form is created. This form allows the festival chairman to create and configure a new music competition.

## ARIA: Create a Competition


Welcome! Please submit information for all of the fields in the form below in order to create a new NNMTA music competition.

**Competition Name**

**Competition Start Date**

**Competition End Date**

◀ Mar ▼ 2016 ▼ ▶

SU	MO	TU	WE	TH	FR	SA
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Illustration 2: Sample competition creation form date selection. One of the configuration options is the competition start and end dates. The festival chairman can enter the date in text form by clicking in the box, or the chairman can select the date by clicking the calendar icon to the right of the box.

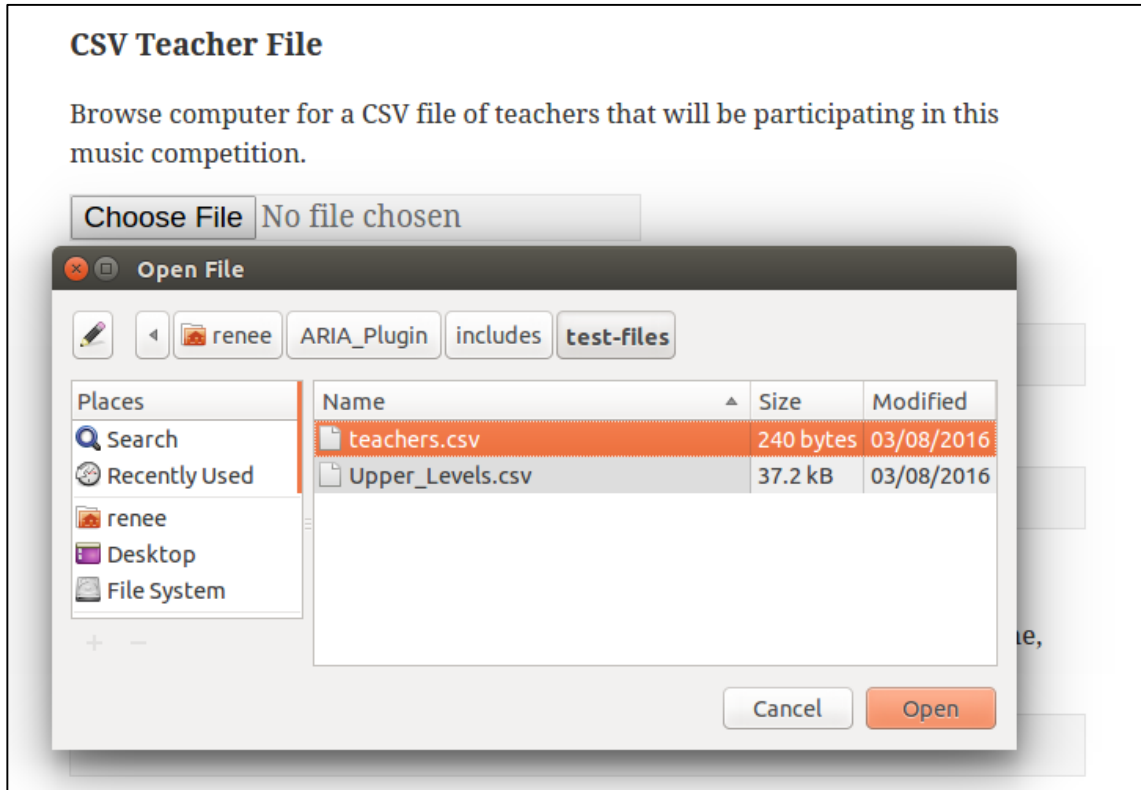


Illustration 3: Sample teacher information upload. When creating a competition, the chairman can select a list of music teachers with their emails prepared in CSV format. By selecting the "Choose File" button, the chairman can browse their computer for this file, which will be added to the data list of teachers participating in the competition.

**Volunteer Time Options for Teachers**

e.g. Saturday (10am-4pm), Either Saturday or Sunday, etc.

Saturday Morning

Saturday Afternoon

Sunday Morning

Illustration 4: Sample time entering for competition creation. The chairman can create options for the teachers to sign up for. These options can be added by typing their names. To add another option, the "+" is clicked. To delete an option, the "-" is clicked. This allows the chairman to add as many options as needed for the competition.

## **ARIA: Create a Competition**

Congratulations! A new music competition has been created. The following forms are now available for students and teachers to use for registration:

[My Competition Student Registration](#) was published.

[My Competition Teacher Registration](#) was published.

Illustration 5: Sample form publication. Once the chairman submits the "Create a Competition" form, the confirmation message is displayed. The student and teacher registration forms are created dynamically, added to the list of pages, and published to the WordPress site. The links for these pages are provided to the festival chairman for reference.

## My Competition Student Registration

**Parent Name**

<input type="text" value="Jane"/>	<input type="text" value="Doe"/>
First	Last

**Parent's Email**

**Student Name \***

Please capitalize your child's first and last names and double check the spelling. The way you type the name here is the way it will appear on all awards and in the Command Performance program.

<input type="text" value="Jonny"/>	<input type="text" value="Doe"/>
First	Last

**Student Birthday**




Illustration 6: Sample student registration form. Once the forms are created and published, the public can access the form through the NNMTA website. Teachers can register their children by entering necessary information in the text boxes as shown.

**Piano Teacher's Name \***

Please select your teachers name from the drop-down below.

wesley kepke ▼

wesley kepke  
fred harris  
kyle lee  
renee iinuma  
ernest landrito

**name is not listed, enter name below.**

**al Days (check all available times)**

There is no guarantee that scheduling requests will be honored.

**Saturday**

Sunday

**Preferred Command Performance Time (check all available times)**

Please check the Command Performance time that you prefer in the event that your child receives a superior rating.

**Thursday 5:30**

**Thursday 7:30**

Illustration 7: Sample teacher selection for student registration. Parents can select their child's piano teacher from the dropdown menu which was populated by the filed uploaded by the festival chairman. If the student's teacher is not listed, the parent can enter the teacher's name instead. Additionally, there are other preferences which are selected with checkboxes.

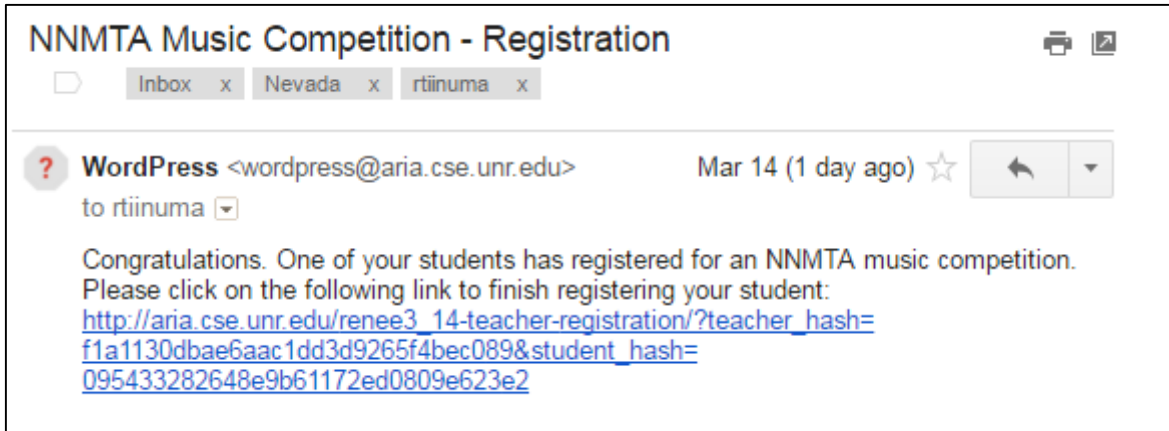


Illustration 8: Sample email to teachers. Once a student is registered, an email is automatically sent to that student's teacher. This email contains a message notifying them that one of their students has finished registering. It also contains a unique link which will take them to the teacher's registration page for that specific student.

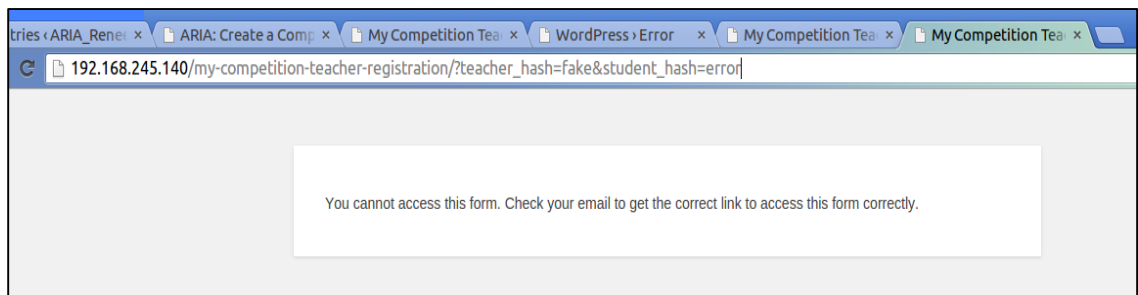
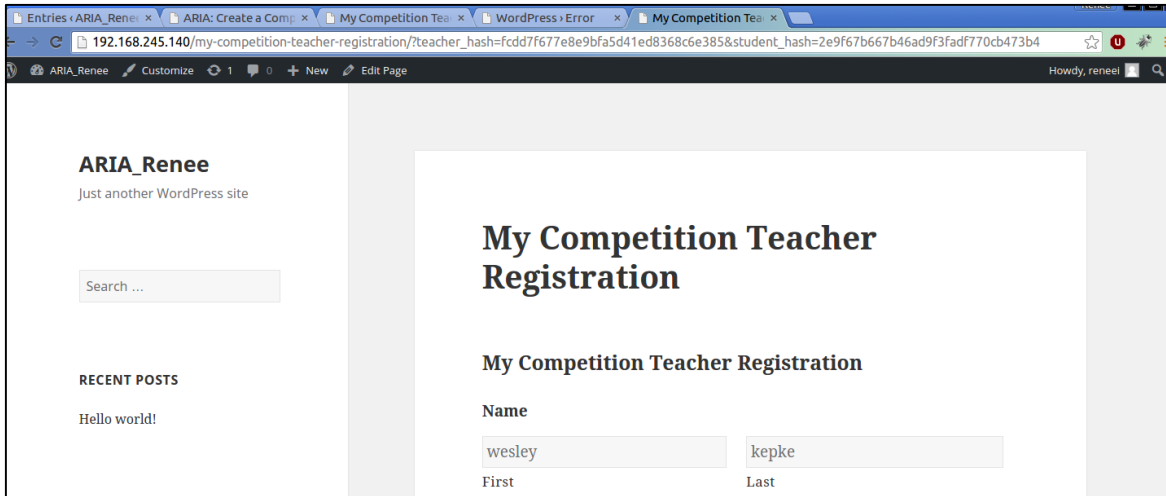


Illustration 9: Sample page of incorrect URL. If someone tries to access the teacher registration page without the correct URL, they are shown an error message and not allowed to register. This allows that only teachers with the correct URL are able to access.



The screenshot shows a web browser window with the following details:

- Browser tabs: ARIA\_Renee, ARIA: Create a Com..., My Competition Tea..., WordPress + Error, My Competition Tea...
- Address bar: 192.168.245.140/my-competition-teacher-registration/?teacher\_hash=fcd7f77e8e9bfa5d41ed8368c6e3858&student\_hash=2e9f67b667b46ad9f3fadf770cb473b4
- Page title: ARIA\_Renee
- Page subtitle: Just another WordPress site
- Search bar: Search ...
- RECENT POSTS: Hello world!
- Main heading: My Competition Teacher Registration
- Form fields: Name (First: wesley, Last: kepke)

Illustration 10: Sample teacher registration page. If the URL is correct, the teachers are taken to the teacher registration page for the competition. Based on the hashes in the URL, the form is prepopulated with important fields such as the teacher's name, student's name, and student's level. This makes it so the teacher does not have to manually enter this information.

**Student Level**  
7 ▾

**Song 1 Period**  
Classical ▾

**Song 1 Composer**  
Select Composer... ▾

**Song 1 Selection**  
▾

**Song 2 Period**  
Baroque ▾  
Baroque  
Romantic  
Contemporary

**Song 2 Selection**  
▾

Illustration 11: Sample period selection. Teachers are responsible for selecting the two songs which their student will be performing. Since they must be from two different periods, if the first song is from the Classical period, then this is removed as an option for the second song's period. This eliminates the aspect of human error of selecting two songs from the same period, which is not allowed.

**Student Level**

7 ▾

**Song 1 Period**

Classical ▾

**Song 1 Composer**

Select Composer... ▾

- Select Composer...
- Albeniz, M.
- Bach, C P E
- Beethoven**
- Cimarosa
- Clementi
- Haydn
- Kirnberger
- Mozart, W.a.
- Reinagle
- Schubert

▾

**Student Level**

7 ▾

**Song 1 Period**

Romantic ▾

**Song 1 Composer**

Select Composer... ▾

- Select Composer...
- Barden
- Beach
- Burgmuller
- Chopin**
- Concone
- Granados
- Grieg
- Macdowell
- Mendelssohn
- Miaskovsky
- Schumann
- Tchaikowsky
- Von Paradis

centage)

Illustration 12 and Illustration 13: Sample composer selection from Classical and Romantic time periods. Once the song's period is chosen, a list of composers from that period are dynamically populated. If the teacher changes the period, the list of composers is repopulated with the composers from the new period. The composers displayed are the composers with only songs approved for the student's level.

**Student Level**

7 ▾

**Song 1 Period**

Romantic ▾

**Song 1 Composer**

Burgmuller ▾

**Song 1 Selection**

Select Song... ▾

Select Song...

**Allegro Agitato In C Minor**

Andante In D

Rondo Ala Turca

**Song 2 Composer**

▾

**Song 2 Selection**

▾

Illustration 14: Sample song selection. Once the song's composer is chosen, the list of songs for that composer are populated. These songs are specific to the student's level, so there will be no error of a student performing a song which is not approved for their level. If the composer is changed, then the list of song options is repopulated with songs for the new composer.

SAVE SCHEDULE	
Saturday	Sunday
<p>Timeblock # 1</p> <p>Section #1</p> <ul style="list-style-type: none"> <li>• Start Time: 9:00 AM</li> <li>• Room: 1</li> <li>• Judge(s): TYPE IN JUDGE(S)</li> <li>• Proctor(s): TYPE IN PROCTOR(S)</li> <li>• Door Guard: TYPE IN DOOR GUARD</li> <li>• Number of Students: 11</li> <li>• Student Skill Levels: 5, 6, 8, 11</li> <li>• Section Type: Traditional/Non-Competitive/Command</li> <li>• Total Play Time: 36 minutes</li> </ul> <p>• Student #1</p> <ul style="list-style-type: none"> <li>• Student Name: Kyle Kim</li> <li>• Student Skill Level: 5</li> <li>• Student Play Time: 5 minutes</li> <li>• Song #1: Mazurka, G minor</li> <li>• Song #2: Solfeggietto</li> </ul> <p>• Student #2</p> <ul style="list-style-type: none"> <li>• Student Name: August Reyman</li> <li>• Student Skill Level: 5</li> <li>• Student Play Time: 5 minutes</li> <li>• Song #1: Little Prelude F Major</li> <li>• Song #2: Knecht Rupert</li> </ul>	<p>Timeblock # 1</p> <p>Section #1</p> <ul style="list-style-type: none"> <li>• Start Time: 9:00 AM</li> <li>• Room: 1</li> <li>• Judge(s): TYPE IN JUDGE(S)</li> <li>• Proctor(s): TYPE IN PROCTOR(S)</li> <li>• Door Guard: TYPE IN DOOR GUARD</li> <li>• Number of Students: 9</li> <li>• Student Skill Levels: 5, 6, 7</li> <li>• Section Type: Traditional/Non-Competitive/Command</li> <li>• Total Play Time: 36 minutes</li> </ul> <p>• Student #1</p> <ul style="list-style-type: none"> <li>• Student Name: Angelo Carlini</li> <li>• Student Skill Level: 5</li> <li>• Student Play Time: 5 minutes</li> <li>• Song #1: Implosion</li> <li>• Song #2: Sonatina C Major 1st-Spirito</li> </ul> <p>• Student #2</p> <ul style="list-style-type: none"> <li>• Student Name: Benjamin Cooke</li> <li>• Student Skill Level: 5</li> <li>• Student Play Time: 4 minutes</li> <li>• Song #1: Sonatina No. 1 C Major</li> <li>• Song #2: Whirlwind, 24CP, p. 10</li> </ul>

Illustration 15: Sample schedule. After registration has completed, the festival chairman is able to enter certain parameters for scheduling. This includes the number of time blocks for performance and the number of concurrent session happening at a time. After specifying these details, a schedule of students is automatically generated. This schedule can be easily modified. Additionally, the festival chairman can specify the judges and proctors for each session. This schedule can be recreated with different parameters until the ideal schedule is generated.

## Reno Youth Music Festival 2016

### Sunday 1:00 PM, Room 1

Judge: Sophia Gilmsen-Univ. of Texas at Austin

**Ernest Landrito**                      \_\_\_Sw/D \_\_\_S \_\_\_E \_\_\_NA \_\_\_W

\_\_\_ Bach, J.S.

Bouree II from French Overture

\_\_\_ Alexander

Jumpin' the Ivories

**Wesley Kepke**                      \_\_\_Sw/D \_\_\_S \_\_\_E \_\_\_NA \_\_\_W

\_\_\_ Bach, J.S.

Moonlight Sonata

\_\_\_ Alexander

Jumpin' the Ivories

**Renee Inuma**                      \_\_\_Sw/D \_\_\_S \_\_\_E \_\_\_NA \_\_\_W

\_\_\_ Bach, J.S.

Bouree II from French Overture

\_\_\_ Alexander

Jumpin' the Ivories

**Kyle Lee**                              \_\_\_Sw/D \_\_\_S \_\_\_E \_\_\_NA \_\_\_W

\_\_\_ Bach, J.S.

Bouree II from French Overture

\_\_\_ Alexander

Jumpin' the Ivories

Illustration 16: Sample document. After the schedule is generated, a number of documents required for the competition can be automatically generated. The document generator takes in information from the scheduler and generates the required paperwork for each session and student. These documents are produced in RTF format so they can be easily modified by the festival chairman if necessary.

## *CONCLUSION*

---

By simplifying and automating almost every aspect of competition management, the work which used to take the NNMTA multiple days can now be completed in just a few clicks. ARIA's functionality is customized to the needs of the NNMTA. The registration process, schedule creation, and document generation capabilities of ARIA effectively reduce the amount of time, money, and paper previously spent to on NNMTA's annual competitions. ARIA has been supporting the Reno Youth Music Festival since April 2016. With the ease of use, generalizability, and scalability of ARIA, it can continue to be used for many years to come.

---

**BIBLIOGRAPHY**

- Genius, R. (2016). GravityForms (Version 1.9.1). WordPress.org: Rocketgenius, Inc.
- Hudson, P. (2015). Working with RTF.
- Jayapandian, M., & Jagadish, H. V. (2008a). Automated creation of a forms-based database query interface. *Proc. VLDB Endow.*, 1(1), 695-709. doi:10.14778/1453856.1453932
- Jayapandian, M., & Jagadish, H. V. (2008b). *Expressive query specification through form customization*. Paper presented at the Proceedings of the 11th international conference on Extending database technology: Advances in database technology, Nantes, France.
- Marcus. Events Manager (Version 5.6.3). WordPress.org: WordPress.
- N-Media. (2014). Simple User Registration (Version 1.8). WordPress.org: WordPress.
- Registrationmagic. (2015). RegistrationMagic-Custom Registration Forms (Version 2.5.4). WordPress.org: Registrationmagic.
- Rosenman, M. A. (1996). A Growth Model for Form Generation Using a Hierarchical Evolutionary Approach. *Computer-Aided Civil and Infrastructure Engineering*, 11(3), 163-174. doi:10.1111/j.1467-8667.1996.tb00320.x
- Williams, B., Damstra, D., & Stern, H. (2015). *Professional WordPress : design and development* (pp. 1 volume). Retrieved from Full-text version via Safari (UNR users only)  
<http://unr.idm.oclc.org/login?url=http://proquest.safaribooksonline.com/?uiCode=unr&xmlId=9781118987247>
- WordPress. Writing a Plugin. *Creating a Plugin*. Retrieved from [https://codex.wordpress.org/Writing\\_a\\_Plugin](https://codex.wordpress.org/Writing_a_Plugin)

---

*APPENDIX A – GLOSSARY OF TERMS*

- Aria
  - A long song accompanying a solo voice usually in an opera. Our project is also called ARIA which stands for Administration, Registration, and Information Assistant.
- Baroque
  - A time period in Europe during 17th and 18th century that followed mannerism. Composers during this time period include Bach, Handel, and Vivaldi. This is a category to select from in the piano competition.
- Chairman
  - The person in charge of scheduling and managing festivals and music competitions.
- Classical
  - A time period from 1750-1820 between Baroque and Romantic time periods. This is a time period to select from in the piano competition.
- Competitive / Non-competitive
  - The competition has both competitive and non-competitive formats. Competitive format means that the student will be given a rating, while the

non-competitive format offers comments and critiques from the judges with no rating.

- Command Performance
  - A performance where the contestants that were rated highest in their division are allowed to perform in front of all the competitors and spectators.
- Composer
  - The person who writes music. The creator of a specific piece of music.
- Contemporary
  - The modern time period of music. This is the category for all recent compositions for students to choose songs from.
- Festival Syllabus
  - The syllabus developed by the NNMTA which contains information about a specific event including performance schedules, guest biographies, organization information, and sponsors.
- Impressionistic
  - The composers from the Impressionistic era spans from 1900-1940 and consists of composers such as Debussy, Ravel, and Griffes. This is a category to select from in the piano competition.
- Judges
  - A group of people that will be ranking the competitors by their skill level and difficulty of music. The judges for the NNMTA are composed of music teachers and college piano professors.

- Key
  - A group of notes upon which the scale is based or the specific pattern of notes that governs a song
- Lower-level
  - The Lower Level Festival is held in May that covers beginner and intermediate levels (1-4).
- Master Class
  - A class given by an expert of a particular discipline to a student of that discipline.
- Moderator
  - A type of user that has permission to oversee the website and make updates and changes.
- Music competition
  - A competition where the participants are judged based off the difficulty of music and how well they perform.
- Music festival
  - A festival oriented towards music that is sometimes presented with a theme such as musical genre, nationality or locality of musicians, or holiday.
- NNMTA
  - The Northern Nevada Music Teacher Association, a community of music teachers in Northern Nevada committed to foster professionalism, maintain a community that provides its members encouraging support, provide a program of continuing education, focused on pedagogical growth,

encourage national certification, organize opportunities for student musical achievement, and encourage networking among the music community, schools, and related organizations.

- Opus Number
  - A separate composition or set of compositions by a particular composer, usually ordered by date of publication.
- PayPal
  - A worldwide online method of payment. PayPal is used as an alternative to traditional paper methods such as checks and money orders. Users will have to use PayPal to pay for competition fees.
- Plugin
  - Programs that can easily be integrated into an existing system and used as part of your Web browser. In the case of ARIA, plugins refer to PHP plugins written for WordPress.
- Proctor
  - The person who manages each section during the competition. This person would make sure the competition runs smoothly as possible.
- Ratings
  - A type of ranking of something or someone that is based off of performance, quality, and standard. For the Reno Youth Music Festival, ratings include “Superior”, “Superior with Distinction”, “Excellent”, and “Needs Attention.”

- Recital
  - Performance of a piece of music played by a solo instrument, singer, or a small group.
- Repertoire syllabus
  - A syllabus which contains over a thousand pieces of music in which the students choose from for the Play-a-thon or the Reno Youth Music Festival. This is developed by the NNMTA board members and is included in the Festival Syllabus.
- Romantic
  - A time period from 1800-1850 characterized by emphasis on emotion and individualism. This is a time period to select from in the competition.
- Scales
  - Any set of musical notes ordered by fundamental frequency or pitch. Piano students are tested on this theory during their competitions.
- Theory Score
  - A score that the students receive after taking a theory exam which is required for festivals.
- Transactions
  - An instance of making a payment in order to enter in the piano competition.
- Upper-level
  - Higher division in a competition which requires performing a piece of music that is high in difficulty.

- WordPress
  - An online, open source website creation tool written in PHP. This is the main platform that ARIA will developed under.