Department of Computer Science & Engineering
University of Nevada, Reno

ARIA: Administration, Registration, and Information Assistant

K.R.E.W.
Renee Iinuma
Wesley Kepke
Ernest Landrito
Kyle Lee

Instructor: Dr. Sergiu Dascalu
Advisor: Dr. Frederick Harris (UNR CSE)
External Advisor: Cindy Harris (NNMTA)
March 18, 2016

Abstract

The goal of ARIA is to create a WordPress plugin that assists the Northern Nevada Music Teachers Association (NNMTA) with the registration, scheduling, and organization of their music competitions. This project is meaningful because it significantly reduces the amount of overhead that NNMTA personnel spend in regards to the preparation of their music competitions. Moreover, this plugin automates tasks and alleviates errors that would otherwise take NNMTA members hours to complete and validate. Once implemented, the NNMTA will be presented with a robust and systematic manner of hosting music competitions for years to come. Furthermore, ARIA is constructed in a generalizable format such that the plugin can be used in situations other than music competitions, thus exposing ARIA's functionality to other sectors.

Introduction

The chief objective of ARIA is to develop a WordPress plugin that alleviates the current hassles that the NNMTA currently undergoes when hosting a music competition. From a high level perspective, ARIA's core components involve music uploading/downloading, competition creation and student/teacher registration, competition scheduling, and the competition document production module.

At the submission date for project part two, music uploading/downloading was complete and development efforts were directed primarily towards the competition creation and student/teacher registration component. Since then, the aforementioned component has undergone significant work and is now close to completion. Moreover, ARIA's developers have invested a notable amount of time finalizing the details of the scheduling algorithm and an initial implementation is now available. The team is beginning to perform stringent tests on the scheduling component using past competition data so that bugs can be identified. Finally, preliminary research into how to produce RTF documents for competitions is underway.

Fortunately, there have been no major changes to ARIA. The requirements for ARIA were crafted with detail by Mrs. Harris and ARIA's developers have been following her documentation closely. Small changes, such as the need to add a new field in the competition creation form and ensuring that this new piece of data flows to it's proper location in the plugin, appear to be the changes that repeatedly occur. Finally, the team has begun practicing a new form of version control that reduces the amount of merge

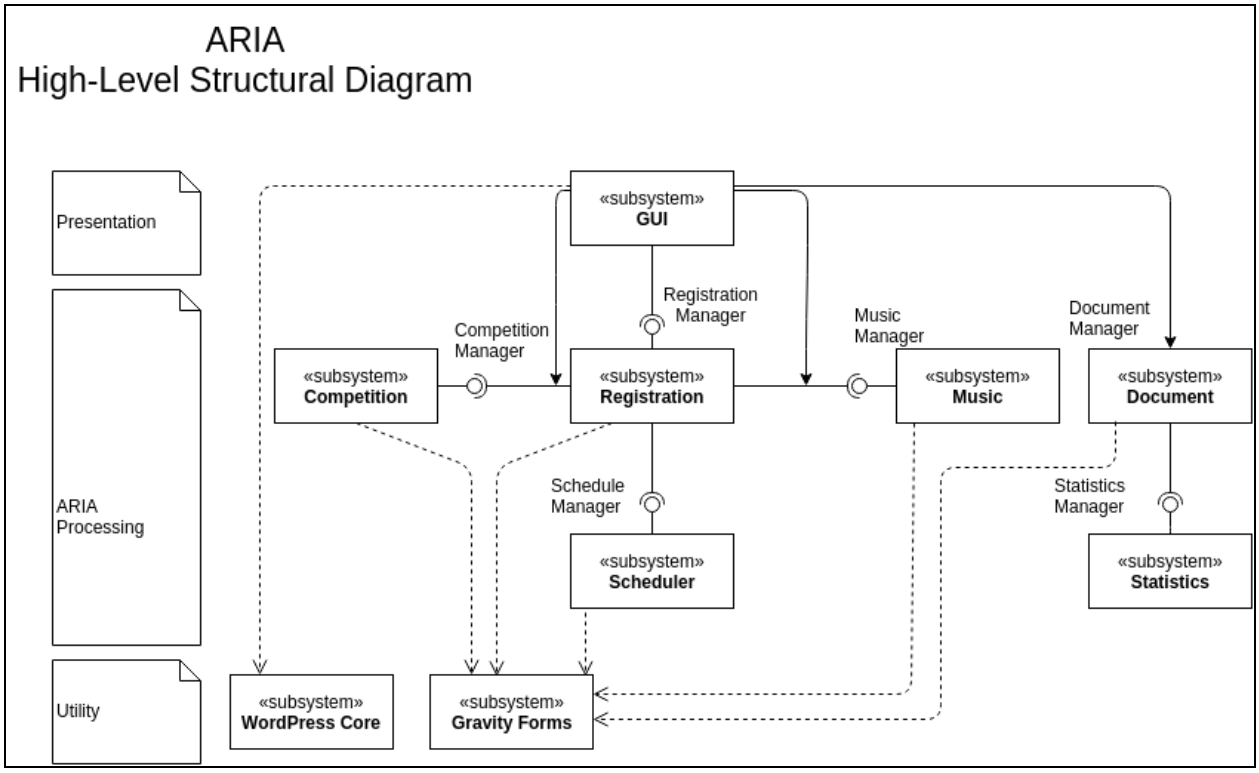
conflicts and allows team members to deviate the work in a much more organized manner.

Perhaps the most difficult challenge that the group has faced in regards to development is working with Gravity Forms, a plugin that serves the purpose of dynamically creating forms that allows users to submit information. The documentation of this rather new plugin is sufficiently lacking in detail and as a result, ARIA's developers had to experiment with the functionality provided by the Gravity Forms API in order to successfully finish the implementation of the competition creation and student/teacher registration component. The previously mentioned phase of experimentation invoked a major setback for ARIA and at the time, there seemed to not be a solution to the problem that ARIA's developers were experiencing. Fortunately, the team found a workaround and was able to implement the solution quickly, thus allowing the team to gain traction and work on ensuing components.

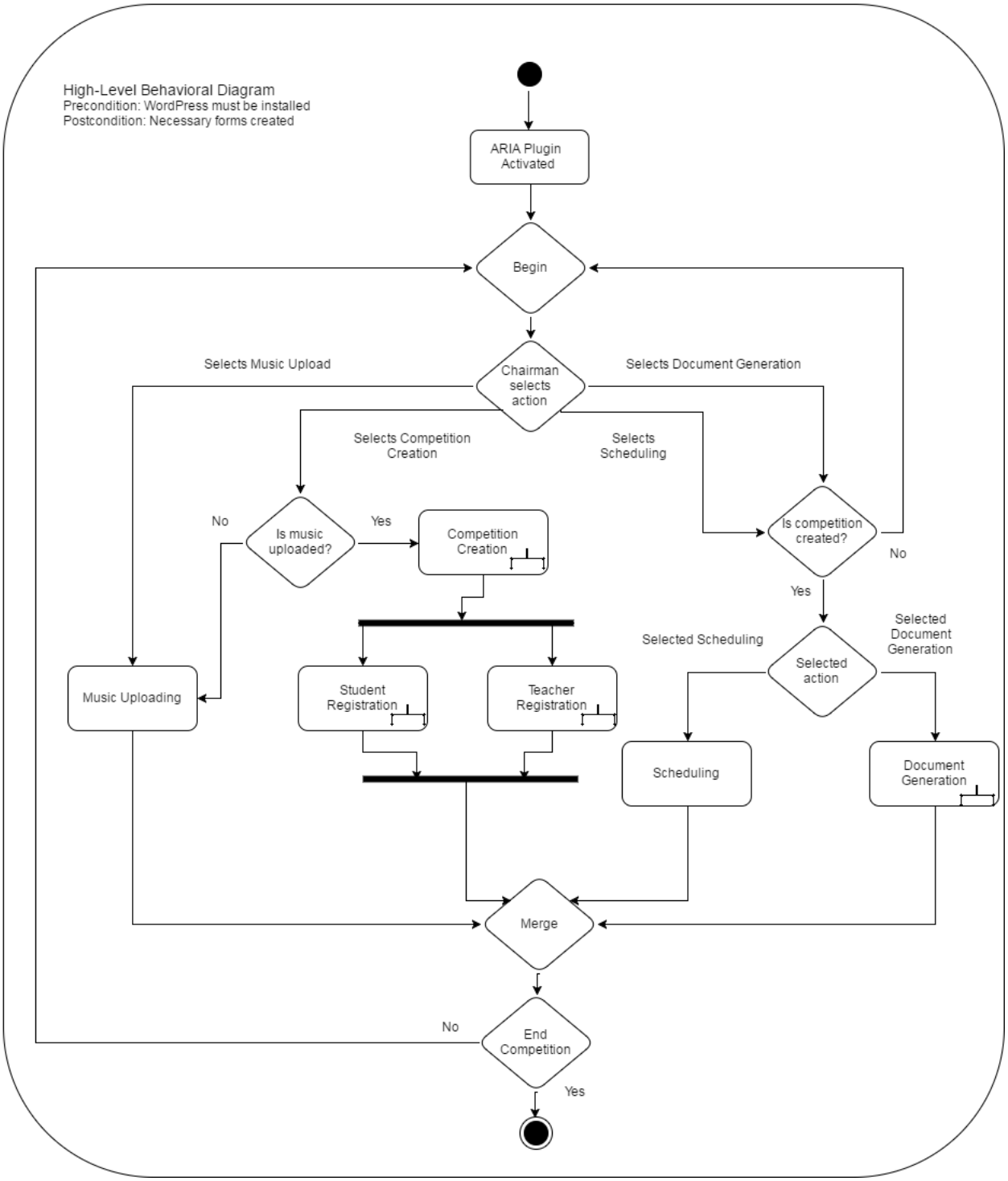
Design Model

Architectural Design

- **High-level structural diagram**
    - The following figure 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.



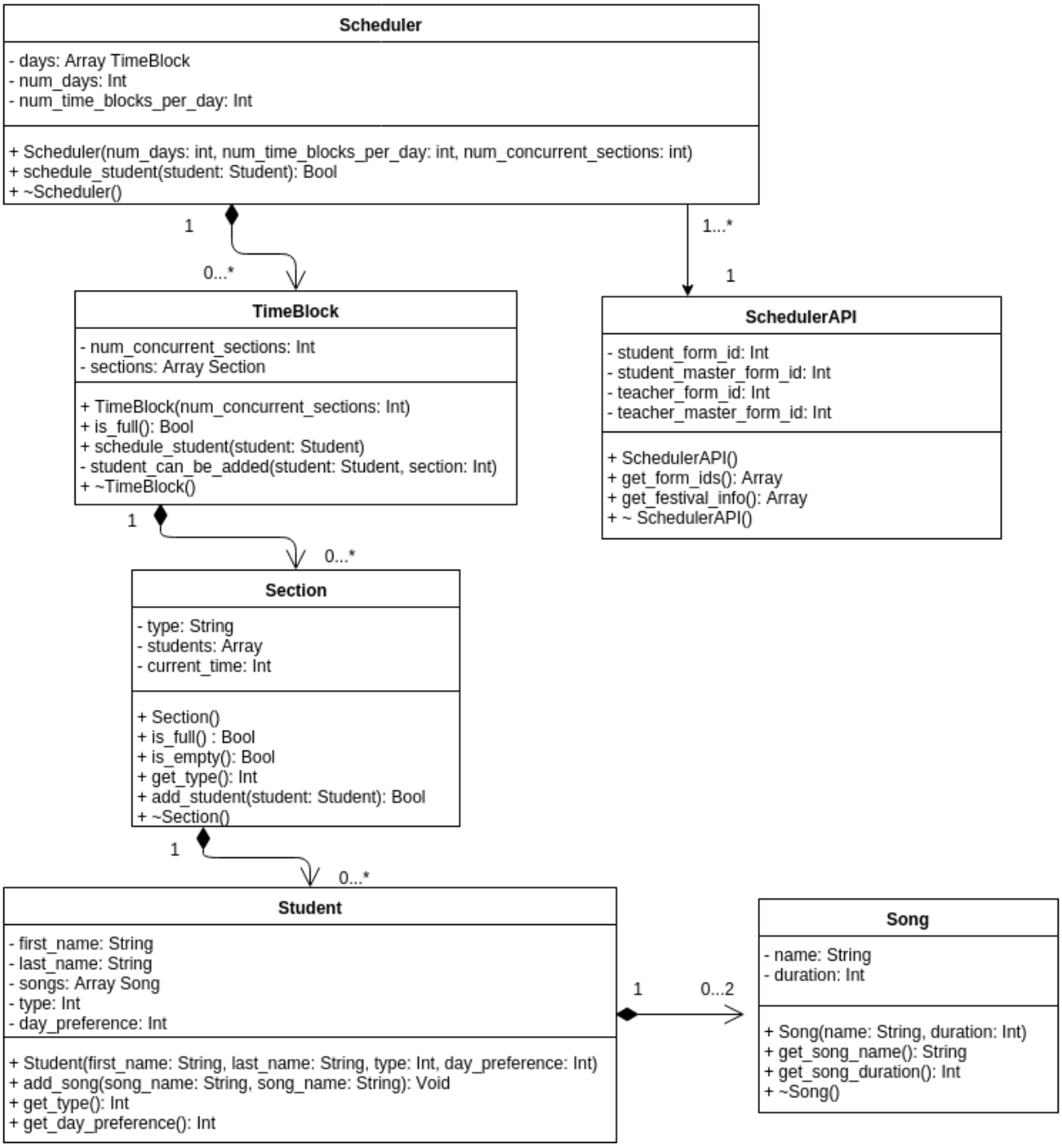
- **High-level behavioral diagram**
    - The following structure depicts the high-level flow of activity of ARIA. Upon activation of the plugin, the chairman is able to select different actions: uploading music, creating a competition, scheduling a competition, or generating documents. The activity 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.


High-Level Behavioral Diagram
Precondition: WordPress must be installed
Postcondition: Necessary forms created
ARIA Plugin Activated
Begin
Chairman selects action
Selects Music Upload
Selects Document Generation
Selects Competition Creation
Selects Scheduling
Is music uploaded?
No
Yes
Competition Creation
Is competition created?
No
Yes
Music Uploading
Student Registration
Teacher Registration
Selected action
Selected Scheduling
Selected Document Generation
Scheduling
Document Generation
Merge
End Competition
No
Yes

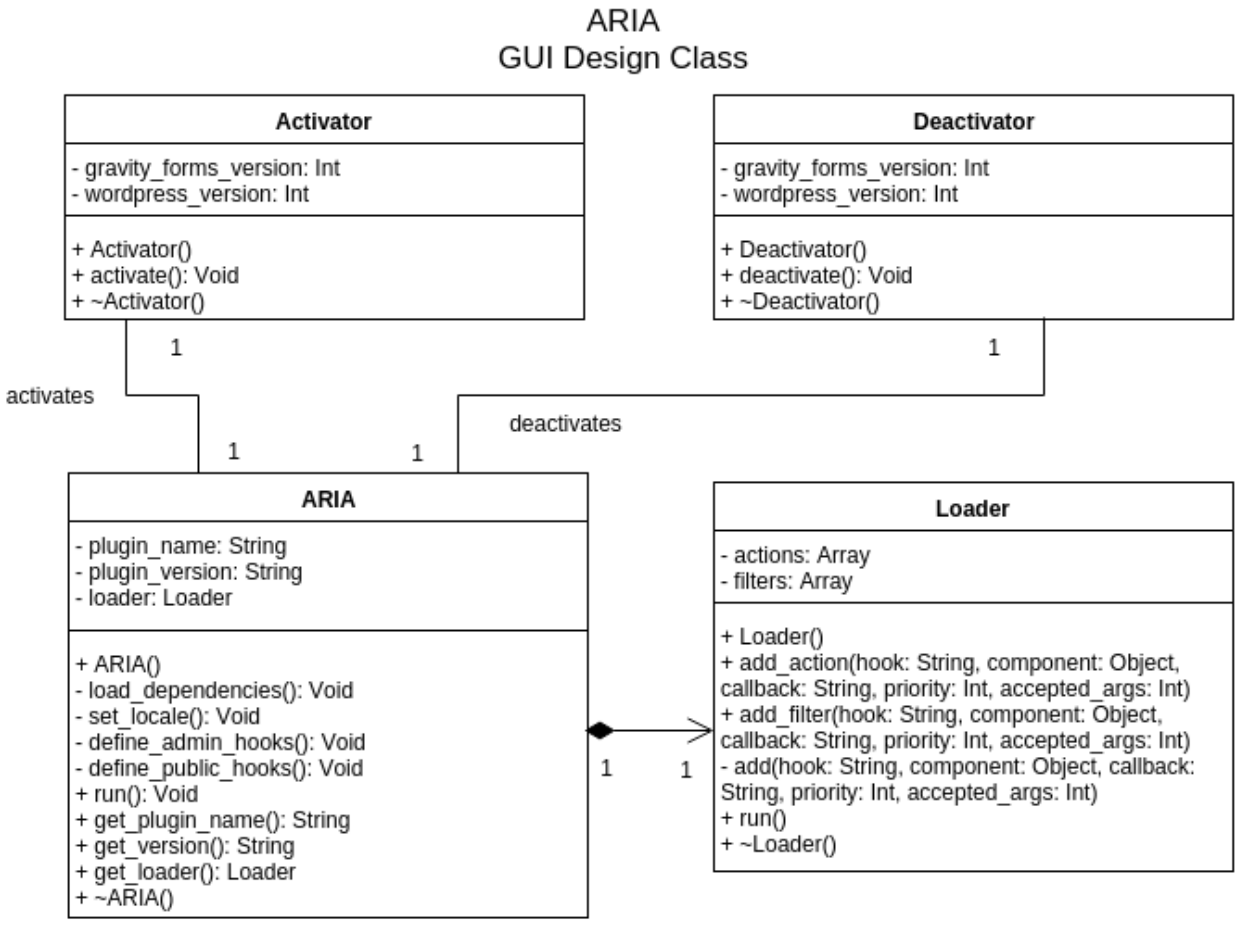
Class Diagrams

- Scheduler Design Class
    - The scheduler design class 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.

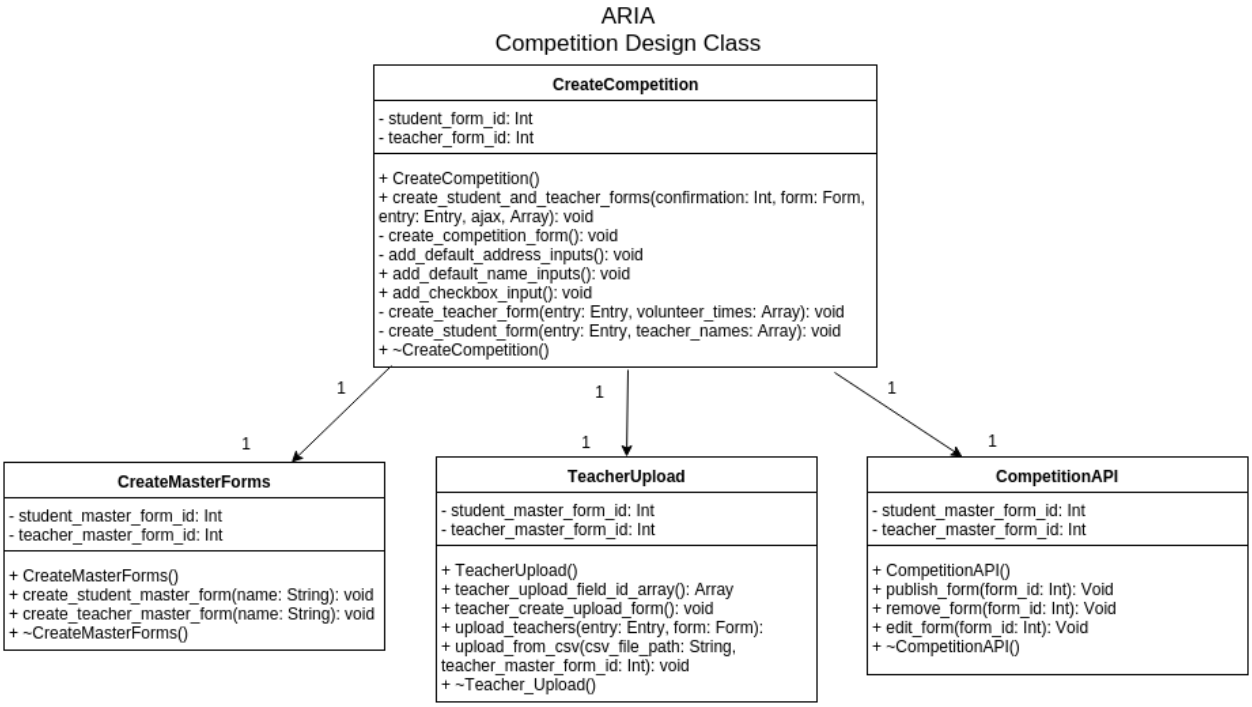
ARIA
Scheduler Design Class



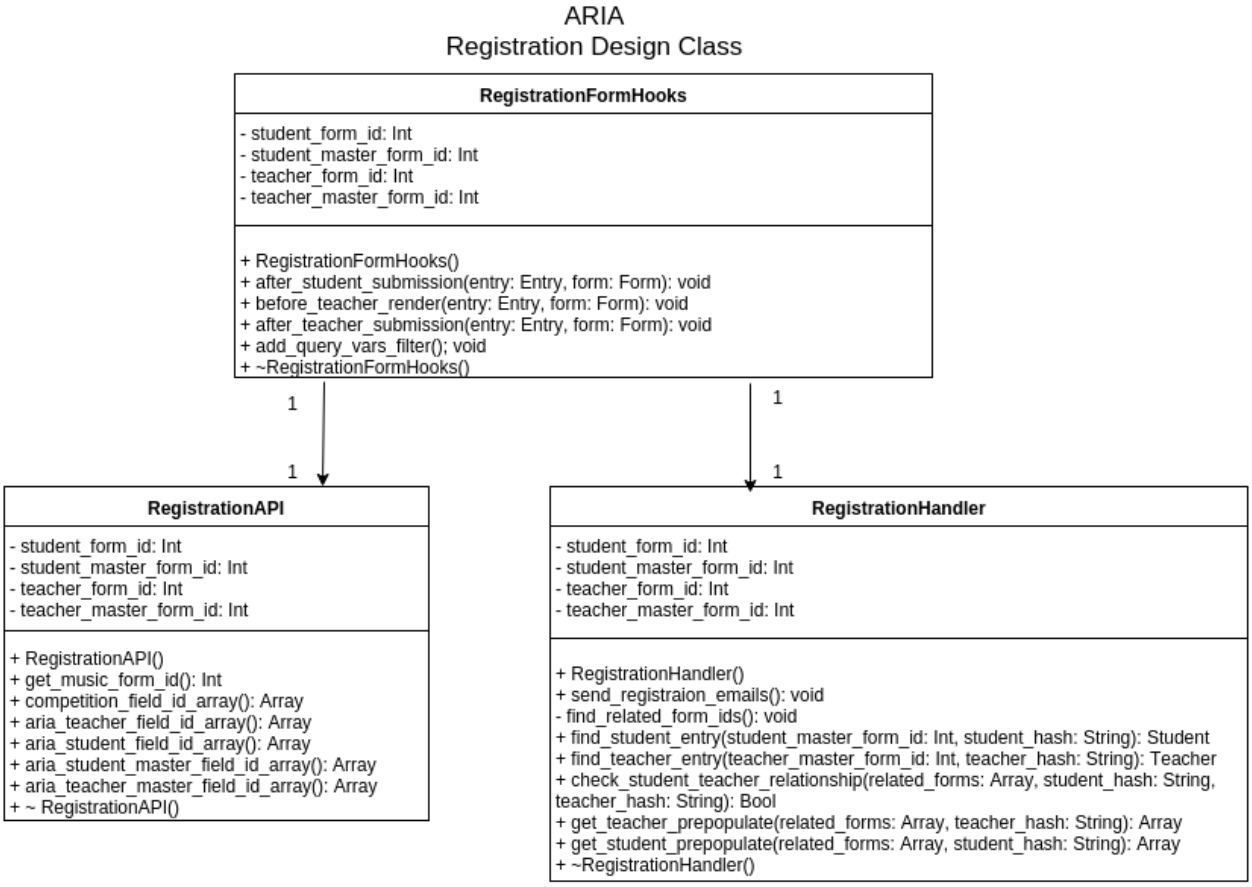
- GUI Design Class
    - The GUI design class relies on “Activator” and “Deactivator” objects in order to correctly perform initialization and deinitialization correctly. The “ARIA” object, which makes use of both the “Activator” and “Deactivator” object, also requires a “Loader” object. The “Loader” 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.



- Competition Design Class
    - The main object in this 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).

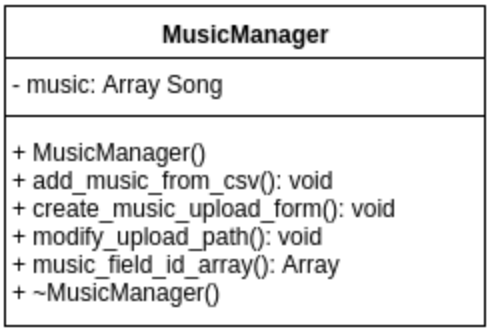


- Registration Design Class
    - The “RegistrationFormHooks” object defines the functionality that is required when students and teachers sign up for an NNMTA music competition. To achieve this functionality, each “RegistrationFormHooks” object communicates with a “RegistrationHandler” object (provides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



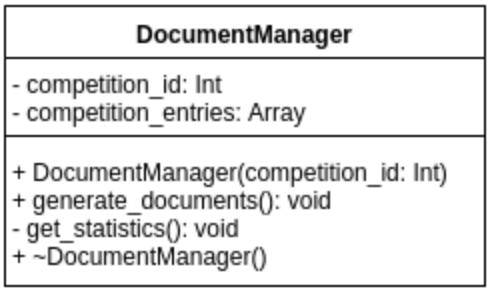
- Music Design Class
    - The “MusicManager” object defines all of the functionality that is needed to upload and modify the music that belongs to the NNMTA.

ARIA
Music Design Class



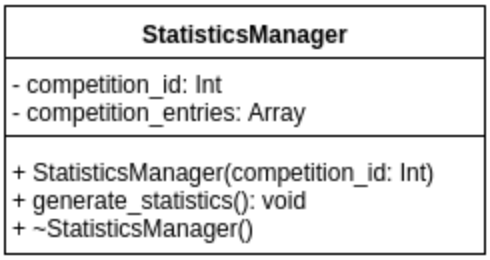
- Document Design Class
    - The “DocumentManager” object defines all of the functionality that is needed to print all of the documents required by the NNTMA on music competition days.

ARIA
Document Design Class



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

ARIA
Statistics Design Class



Program Units

1. Create_competition_activation
    a. 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.
    b. Precondition: none
    c. Postcondition: A new form for creating new music competitions will be created.
    d. Author: Ernest, Wesley
2. Create_teacher_and_student_forms
    a. 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.
    b. Precondition: entry in create competition submitted.
    c. Postcondition: student and teacher forms for a competition is made.
    d. Author: Ernest, Kyle, Wesley
3. Create_competition_form
    a. 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.
    b. Precondition: none
    c. Postcondition: The forms necessary will be created for a competition.
    d. Author: Kyle
4. Add_default_address_inputs
    a. 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.
    b. Precondition: field initialized
    c. Postcondition: inputs will be added to an address field
    d. Author: Kyle
5. Add_default_name_inputs
    a. This function is responsible for adding default name inputs to a name field.

    b. Precondition: field initialized
    c. Postcondition: inputs will be added to a name field
    d. Author: Renee
6. Add_checkbox_input
    a. This function is responsible for adding checkbox inputs to a checkbox field.
    b. Precondition: field initialized
    c. Postcondition: inputs will be added to a checkbox field
    d. Author: Renee
7. Aria_create_teacher_form
    a. 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.
    b. Precondition: a competition is being created.
    c. Postcondition: a form for teacher registration is created.
    d. Author: Ernest
8. Create_student_form
    a. 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.
    b. Precondition: a competition is being created.
    c. Postcondition: a form for student registration is created.
    d. Author: Ernest
9. After_student_submission
    a. 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.
    b. Precondition: a student has submitted their registration.
    c. Postcondition: an entry in teacher database and student database will be created and an email to their piano teacher will be sent.
    d. Author: Wesley, Kyle
10. Before_teacher_render
    a. 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.

b. Precondition: none
c. Postcondition: the form is rendered or the user is navigated to the homepage.
d. Author: Ernest, Renee

11. After_teacher_submission
    a. 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.
    b. Precondition: the teacher is submitting a form.
    c. Postcondition: the corresponding entries in the teacher database and student database will be created.
    d. Author: Ernest, Wesley
12. Add_query_vars_filter
    a. 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.
    b. Precondition: none
    c. Postcondition: Wordpress will be able to detect values in the url.
    d. Author: Ernest
13. Create_student_master_form
    a. 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.
    b. Precondition: a competition is in the process of being created.
    c. Postcondition: A form will be created to hold student information.
    d. Author: Ernest, Renee, Wesley
14. Create_teacher_master_form
    a. 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.
    b. Precondition: a competition is in the process of being created.
    c. Postcondition: A form will be created to hold student information.
    d. Author: Ernest, Renee, Wesley
15. Add_music_from_csv
    a. 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.
    b. Precondition: none
    c. Postcondition: the database will be filled with new music entries.
    d. Author: Kyle, Renee, Wesley
16. Create_music_upload_form
    a. 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.
    b. Precondition: none
    c. Postcondition: the form for adding music to a the database is created.
    d. Author: Kyle, Wesley
17. Create_nnmta_music_form
    a. 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.
    b. Precondition: none
    c. Postcondition: The NNMTA music form will be created.
    d. Author: Renee, Wesley
18. Remove_all_music_from_nnmta_database
    a. 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.
    b. Precondition: none
    c. Postcondition: the music database will be cleared.

    d. Author: Kyle, Wesley
19. Modify_upload_path
    a. 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.
    b. Precondition: none
    c. Postcondition: file uploads will upload to a specified path.
    d. Author: Kyle, Renee
20. Get_create_competition_form_id
    a. 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.
    b. Precondition: none
    c. Postcondition: none
    d. Author: Wesley
21. Get_teacher_upload_form_id
    a. 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.
    b. Precondition: none
    c. Postcondition: none
    d. Author: Renee
22. Get_song_upload_form_id
    a. 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.
    b. Precondition: none
    c. Postcondition: none
    d. Author: Kyle
23. Get_nnmta_database_form_id
    a. 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.
    b. Precondition: none
    c. Postcondition: none
    d. Author: Wesley

24. Send_registration_emails
    a. 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.
    b. Precondition: Student Submitting their registration
    c. Postcondition: Teacher is sent an email.
    d. Author: Wesley
25. Find_related_forms_ids
    a. 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.
    b. Precondition: none
    c. Postcondition: none
    d. Author: Wesley
26. Find_student_entry
    a. 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.
    b. Precondition: none
    c. Postcondition: none
    d. Author: Ernest
27. Find_teacher_entry
    a. 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.
    b. Precondition: none
    c. Postcondition: none
    d. Author: Ernest
28. Check_student_teacher_relationship
    a. This is a function that is used to check if a student is assigned to a teacher.
    b. Precondition: none
    c. Postcondition: none
    d. Author: Ernest
29. Get_teacher_pre_populate

a. 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.
b. Precondition: none
c. Postcondition: none
d. Exception: the teacher doesn't exist.
e. Author: Ernest

30. Get_teacher_pre_populate
    a. 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.
    b. Precondition: none
    c. Postcondition: none
    d. Exception: The student doesn't exist.
    e. Author: Ernest
31. CalculateSig
    a. 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.
    b. Precondition: Crypto script is loaded.
    c. Postcondition: none
    d. Author: Renee
32. Get_songs
    a. 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.
    b. Precondition: Student level is known, Gravity Forms is functioning.
    c. Postcondition: Song information is loaded.
    d. Author: Renee
33. Store_periods
    a. 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.
    b. Precondition: Song periods have been loaded into one of the song's dropdown.
    c. Postcondition: Song periods are stored into an array for later access.
    d. Author: Renee
34. Load_composers
    a. 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.

b. Precondition: Period is selected.
c. Postcondition: Composer dropdown is populated.
d. Author: Renee

35. Load_periods
    a. 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.
    b. Precondition: none
    c. Postcondition: All periods are restored to specified dropdown.
    d. Author: Renee
36. Get_Music_Form_ID
    a. 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.
    b. Precondition: Current competition ID is known.
    c. Postcondition: Music form ID is known.
    d. Author: Renee
37. toTitleCase
    a. 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.
    b. Precondition: none
    c. Postcondition: none
    d. Author: Renee
38. Schedule_student
    a. This function will schedule a student depending on which day they had requested when they registered for a competition.
    b. Precondition: the student has yet to be scheduled for the given competition.
    c. Postcondition: the student has been scheduled according to their request and the student's parents are emailed with the student's schedule time.
    d. Author: Wesley, Kyle
39. Add_student
    a. 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.
    b. Precondition: none

c. Postcondition: the student has been scheduled to a given time section for the competition.
d. Author: Wesley, Kyle

40. Student_can_be_added
    a. 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.
    b. Precondition: none
    c. Postcondition: the student has been given permission or has been rejected from the given time section.
    d. Wesley, Kyle

Detailed Design

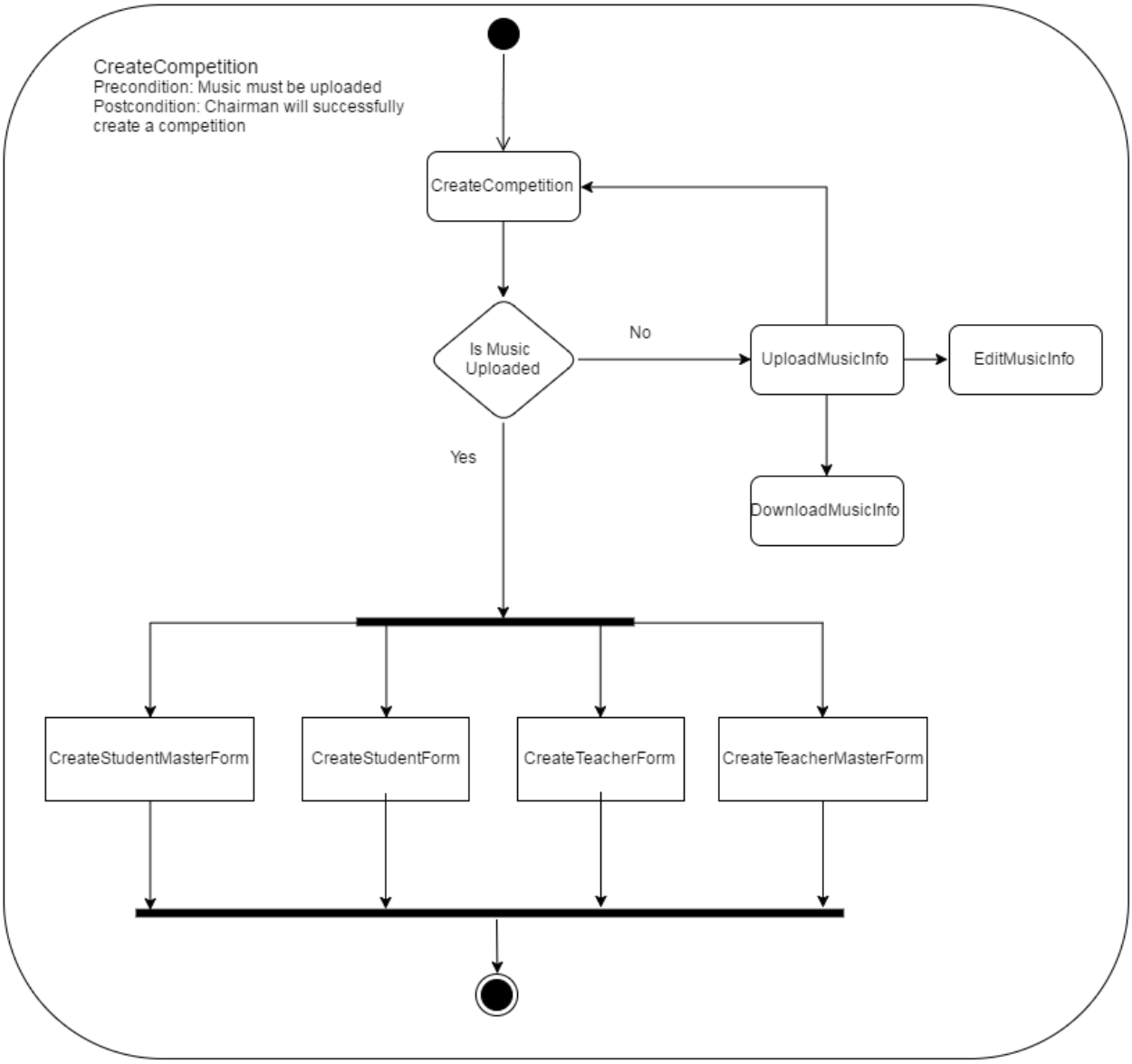


Figure 1: The activity diagram depicts the flow of activity for the process of creating a competition. If music data has not been uploaded, the chairman will be able to upload, edit, and download music. Once music data has been uploaded, the student and teacher forms are created along with the student and teacher master forms. Upon completion of these tasks, the activity terminates.

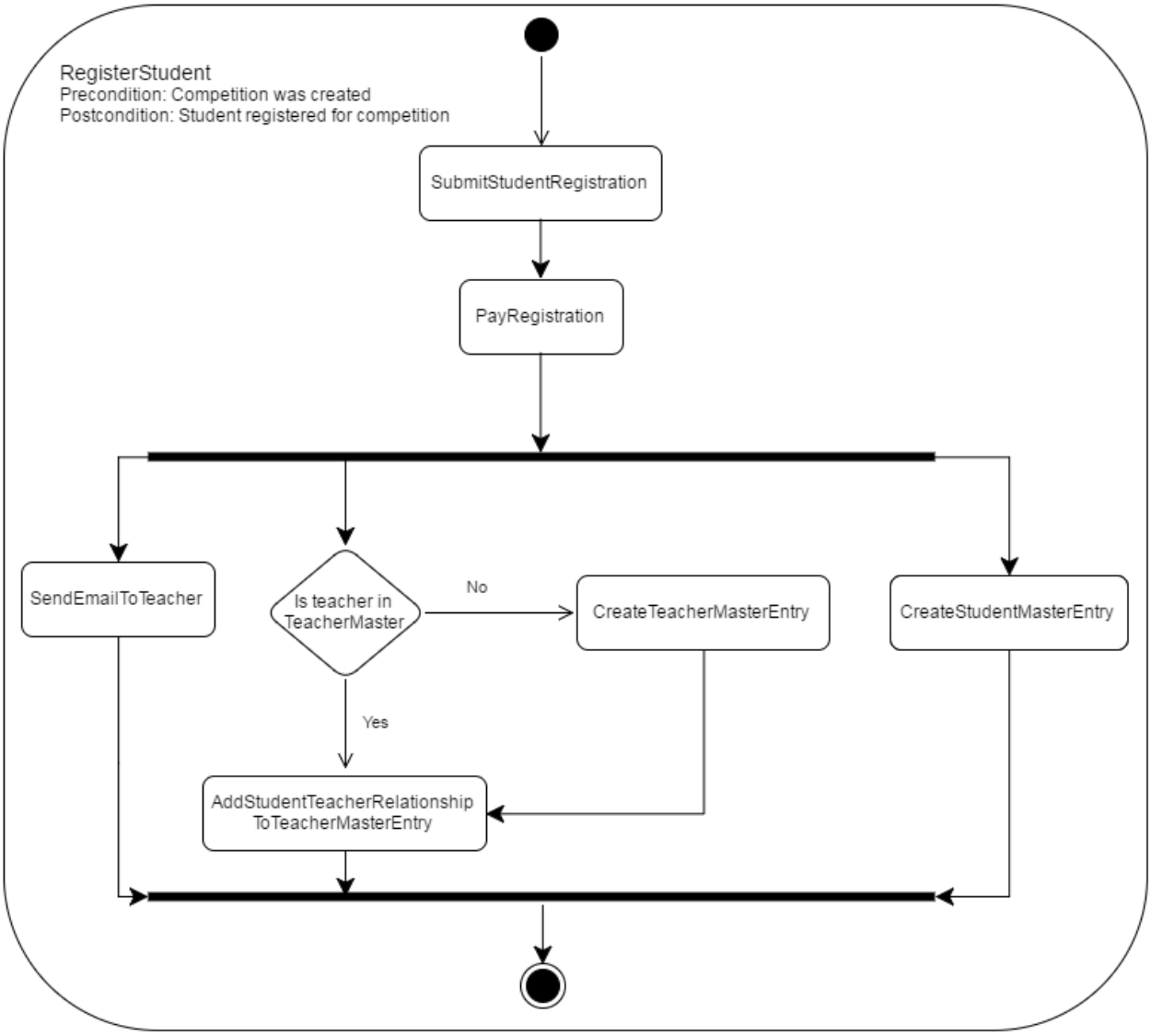


Figure 2: The above activity diagram depicts the student registration process. Once the student submits the registration form, they must pay for the competition. Upon completion of this, an email is sent to the teacher notifying them that one of their students has registered. Additionally, an entry in 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.

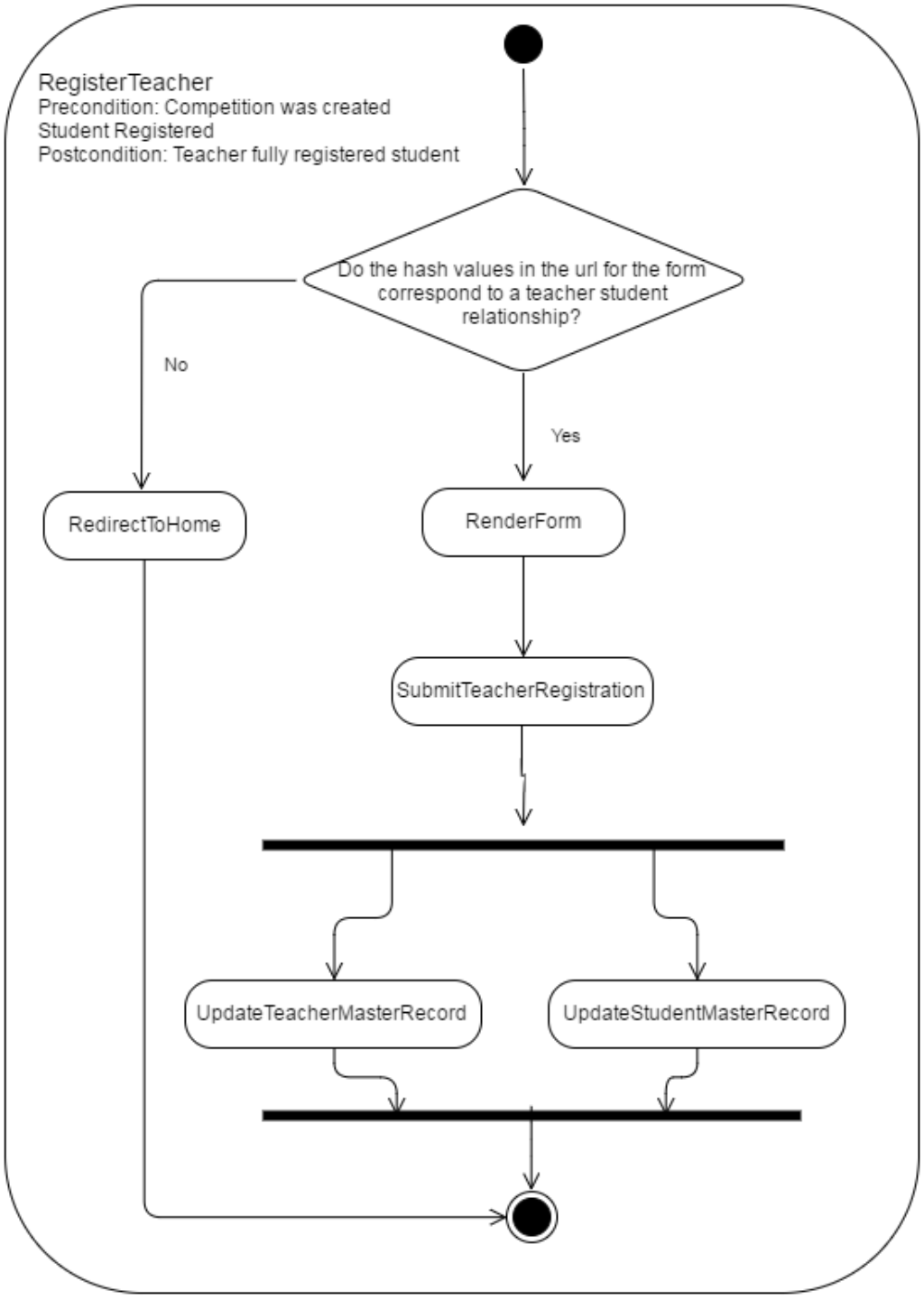


Figure 3: The above activity diagram depicts the teacher registration process. To begin, the hash values are validated. If the hash values are incorrect, then the user is redirected and activity terminates. Otherwise, the form is rendered with some pre-populated values. After the teacher submits the form, the entries in the teacher and student master database are updated.

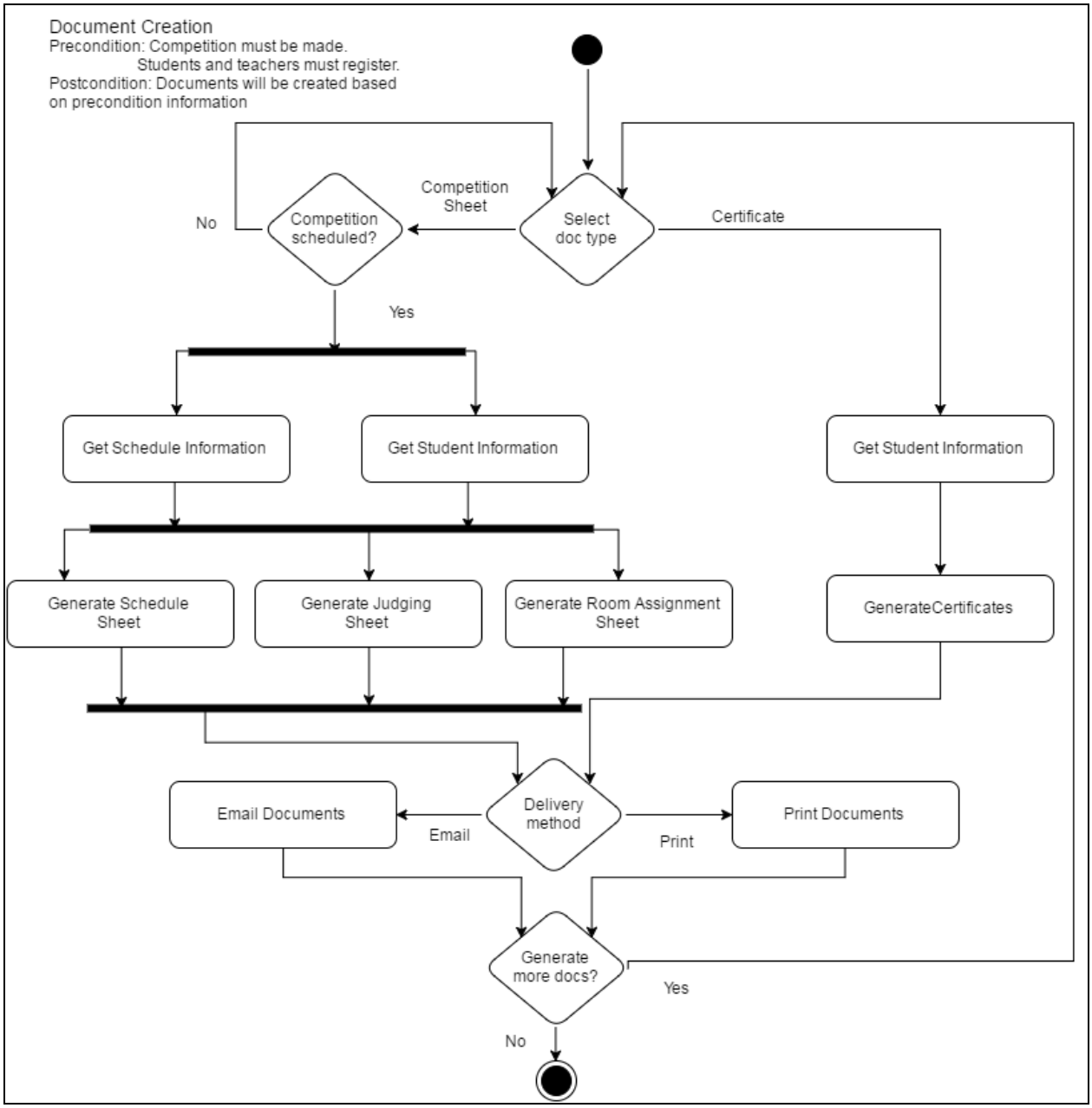


Figure 4: The activity diagram above depicts the process of document generation. The chairman is able to select which type of document to generate. If they select a competition sheet, then a competition must already have been scheduled. If so, the information for the competition schedule and students 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.

Data Design

- 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
        - Judging option

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 figures below.

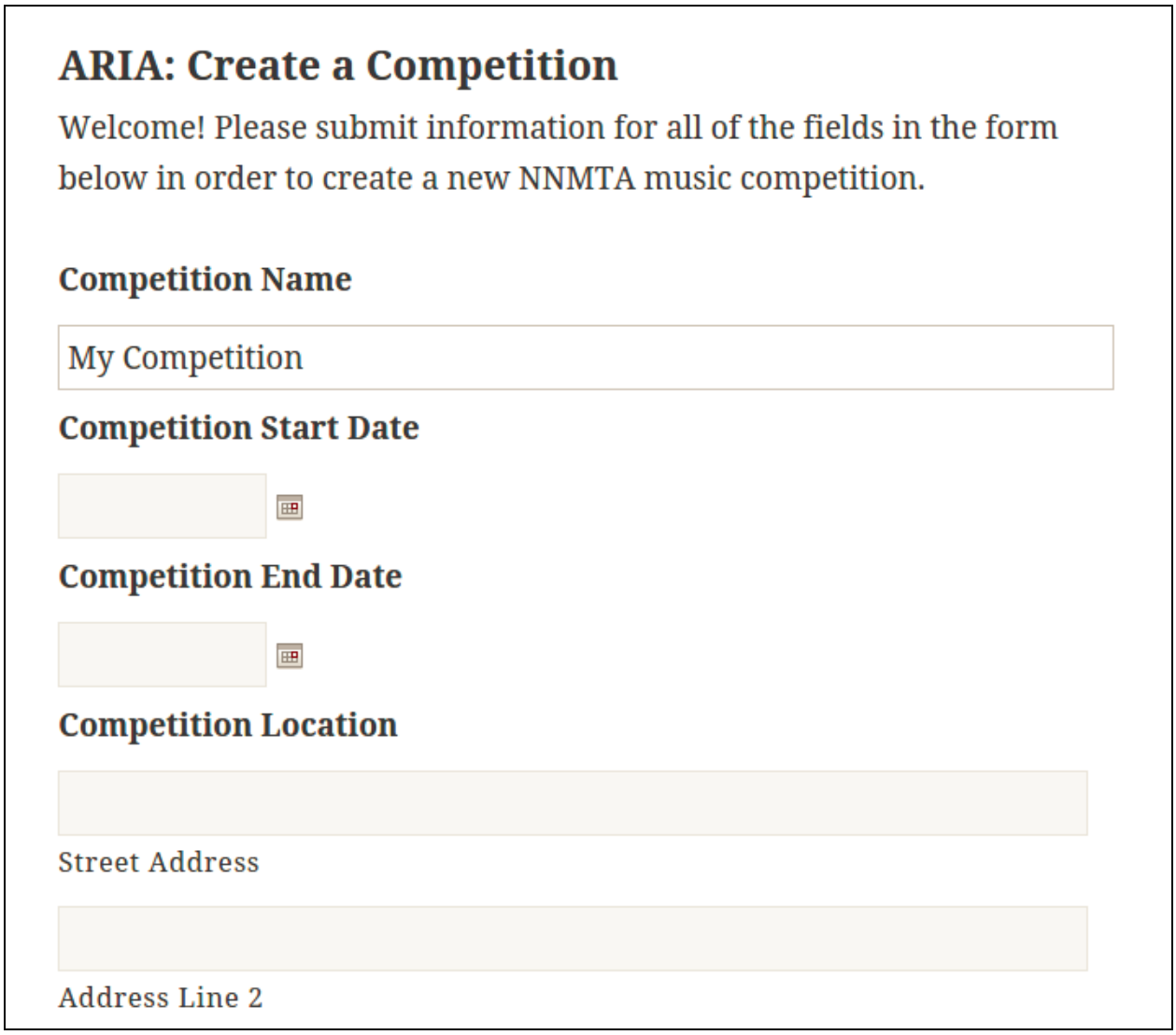


Figure 5: When ARIA is activated, a WordPress page with this form is created. This form allows the festival chairman to create and configure a new music competition.

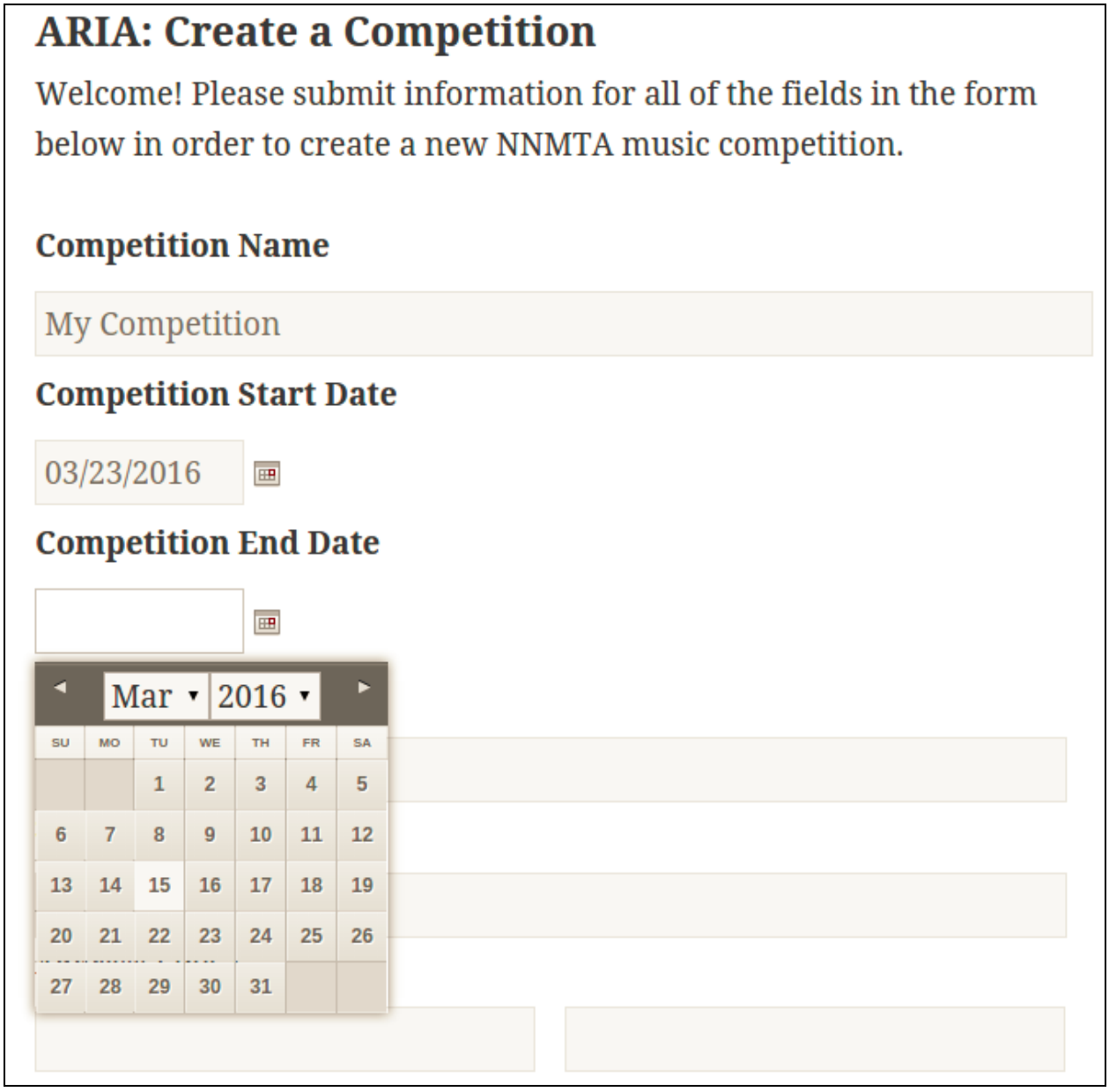


Figure 6: One of the configuration options is the competition start and end dates. The festival chairman can enter the date in text form by clicking in the box, or the chairman can select the date by clicking the calendar icon to the right of the box.

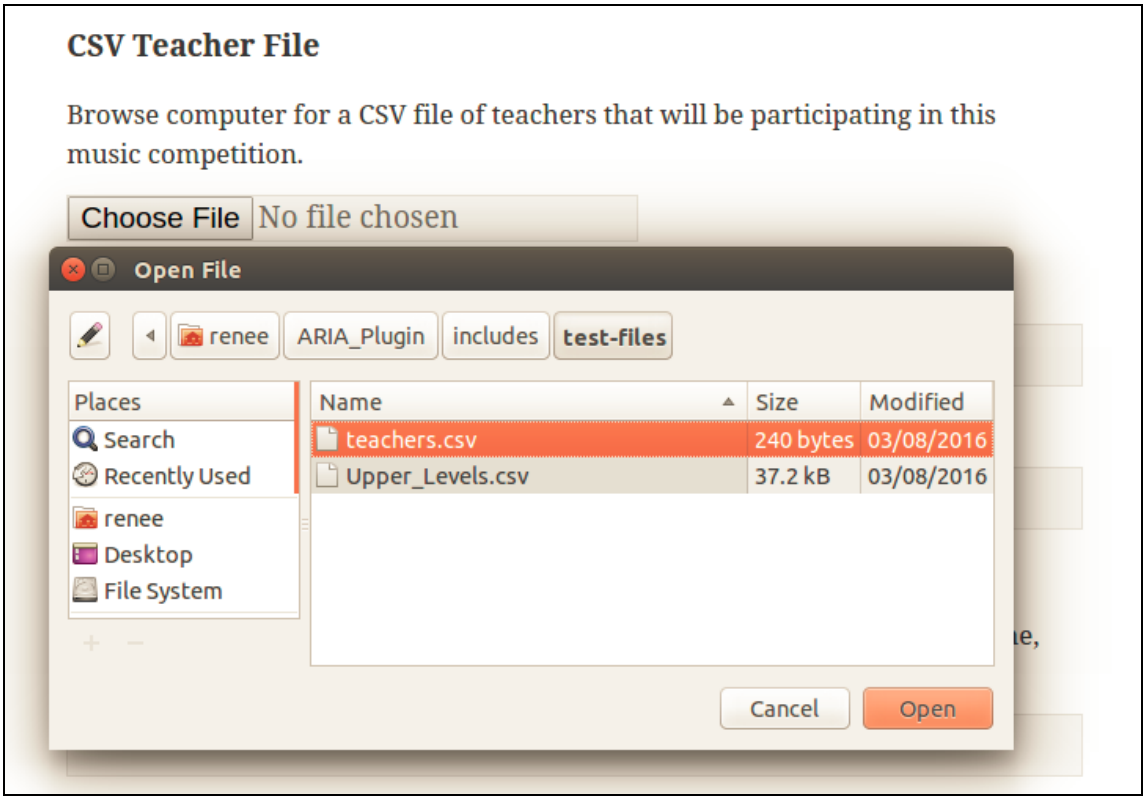


Figure 7: When creating a competition, the chairman can select a list of music teachers with their emails prepared in CSV format. By selecting the "Choose File" button, the chairman can browse their computer for this file, which will be added to the data list of teachers participating in the competition.

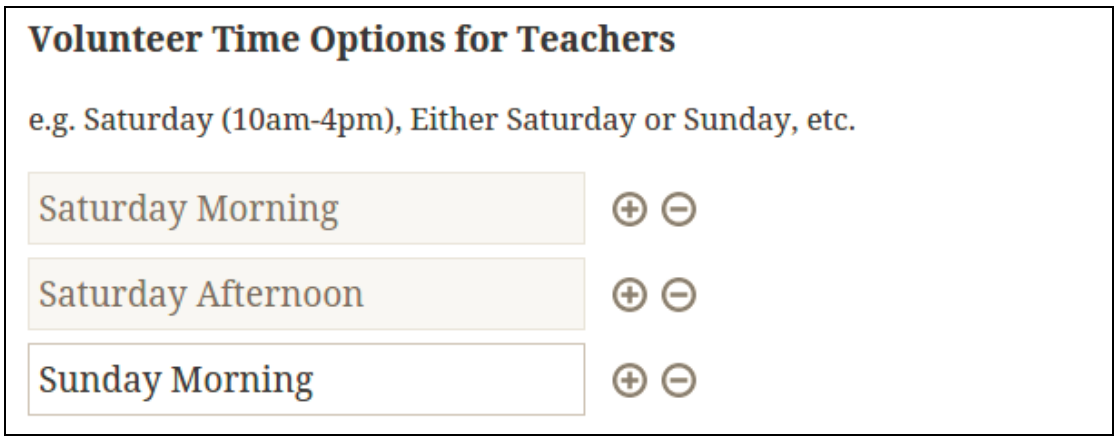


Figure 8: The chairman can create options for the teachers to sign up for. These options can be added by typing their names. To add another option, the "+" is clicked. To delete an option, the "-" is clicked. This allows the chairman to add as many options as needed for the competition.

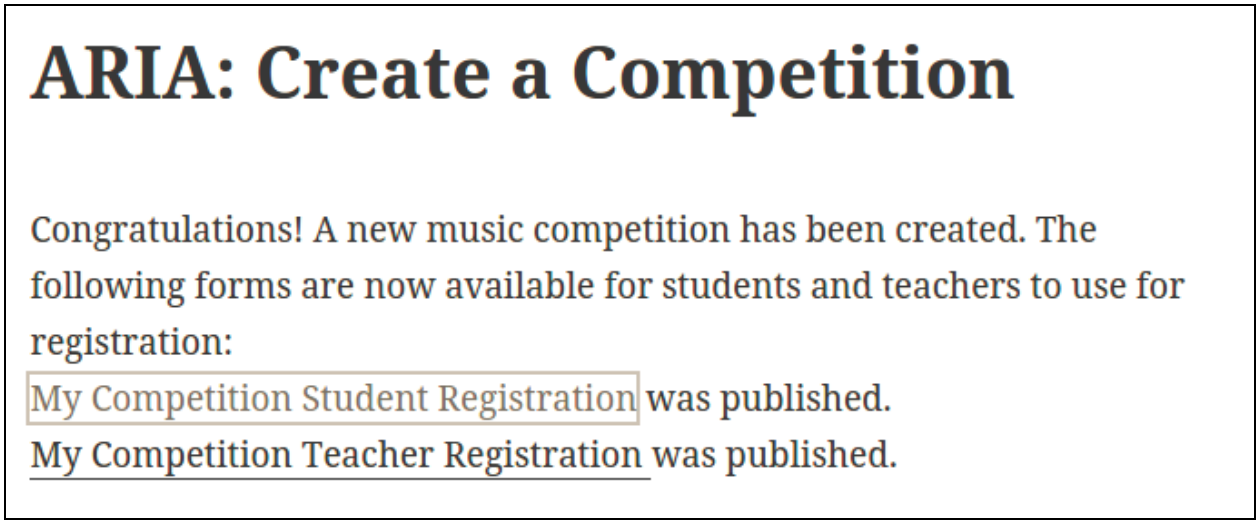


Figure 9: Once the chairman submits the "Create a Competition" form, the confirmation message is displayed. The student and teacher registration forms are created dynamically, added to the list of pages, and published to the WordPress site. The links for these pages are provided to the festival chairman for reference.

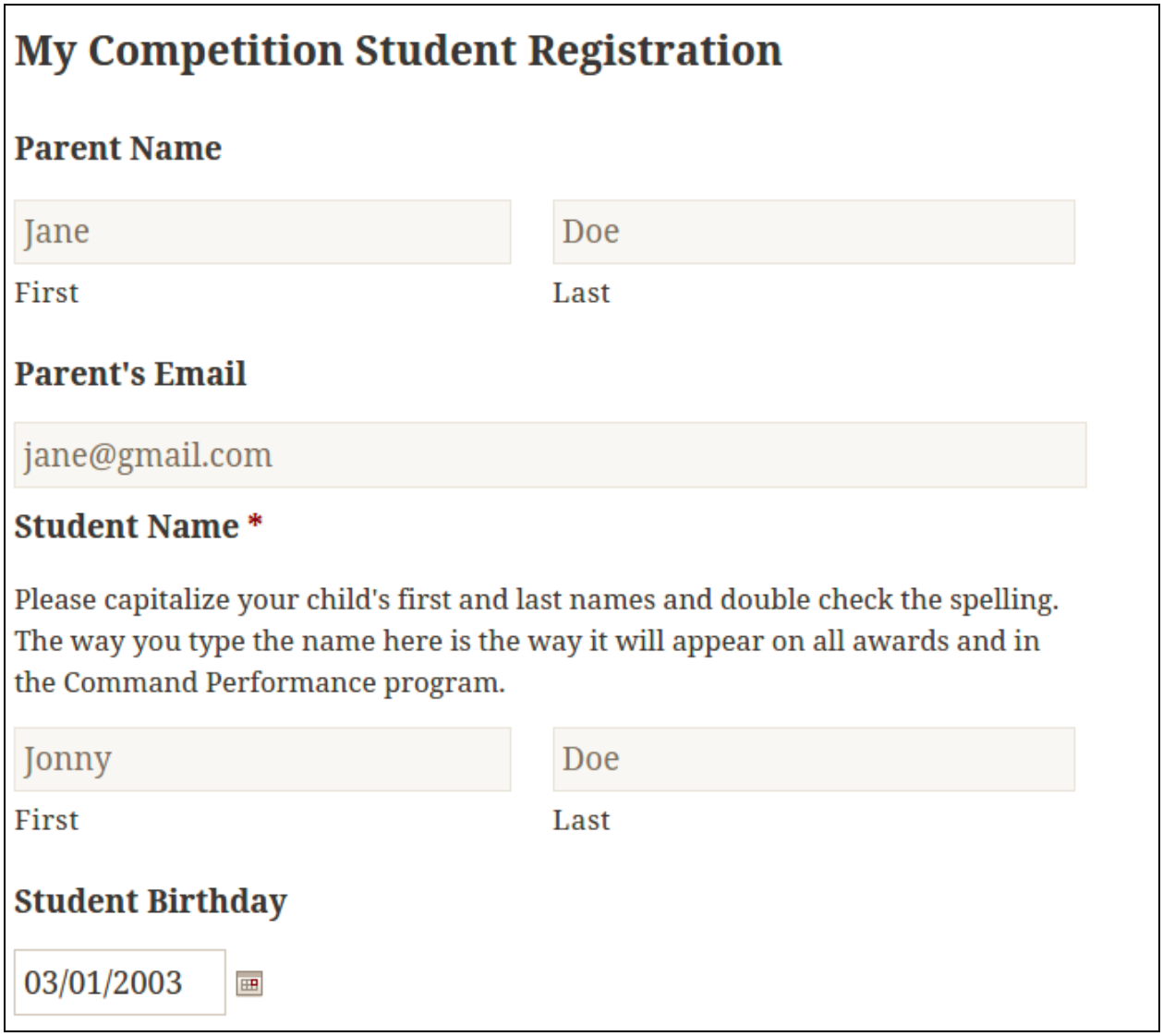


Figure 10: Once the forms are created and published, the public can access the form through the NNMTA website. Teachers can register their children by entering necessary information in the text boxes as shown.

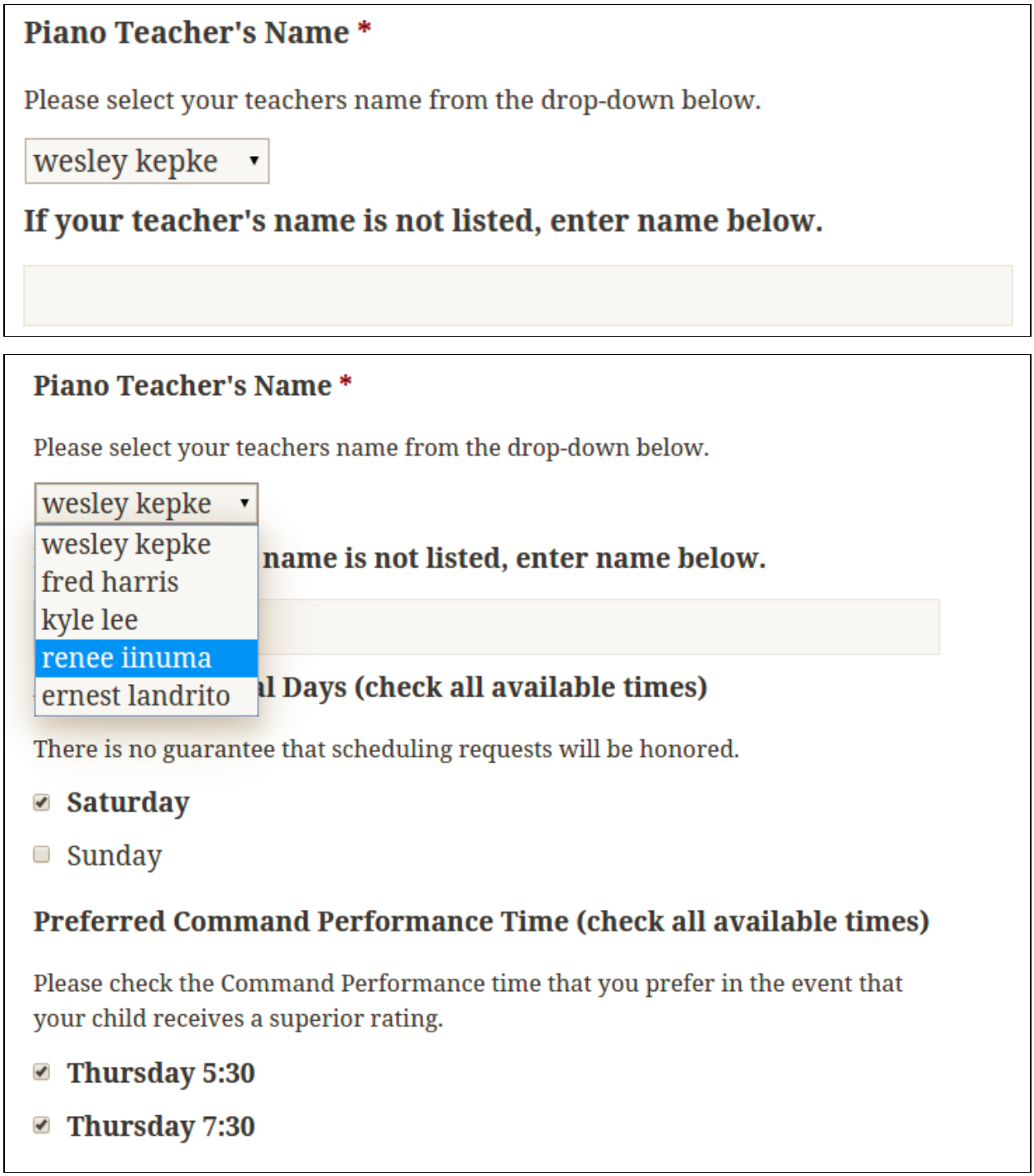


Figure 11.1 and 11.2: Parents can select their child's piano teacher from the dropdown menu which was populated by the filed uploaded by the festival chairman. As shown in Figure X.1, if the student's teacher is not listed, the parent can enter the teacher's name instead. Additionally, there are other preferences which are selected with checkboxes.

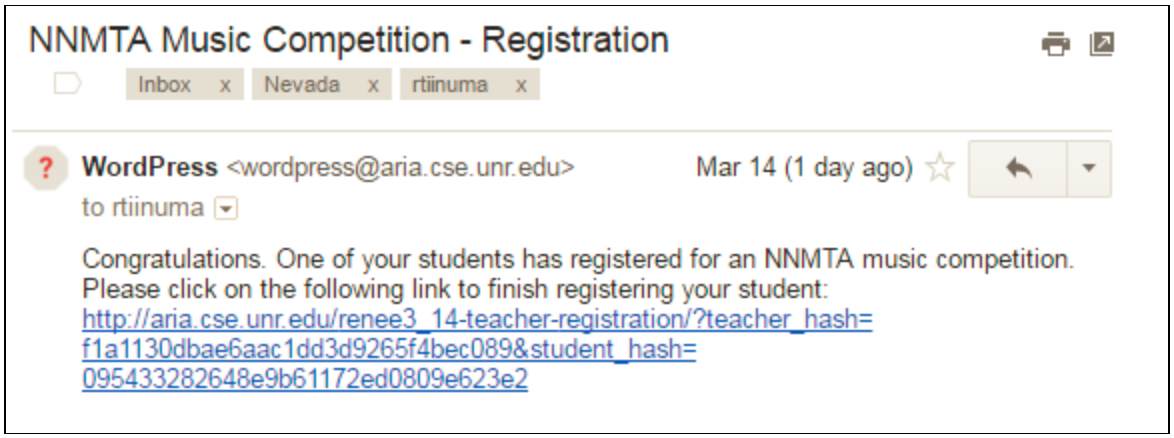


Figure 12: Once a student is registered, an email is automatically sent to that student's teacher. This email contains a message notifying them that one of their students has finished registering. It also contains a unique link which will take them to the teacher's registration page for that specific student.

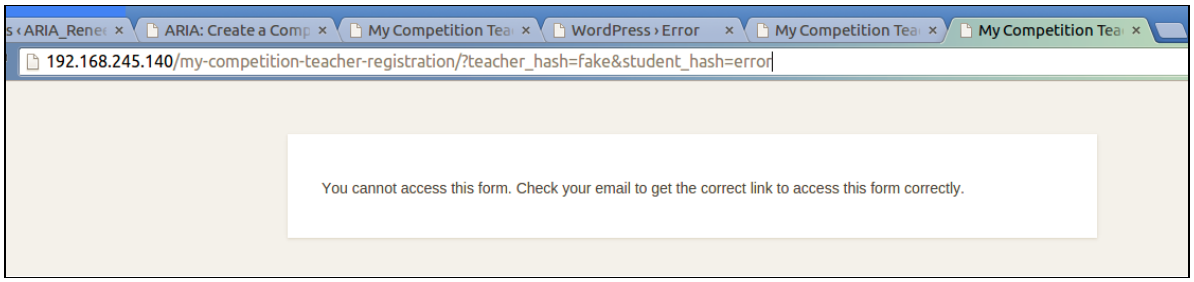


Figure 13: If someone tries to access the teacher registration page without the correct URL, they are shown an error message and not allowed to register. This allows that only teachers with the correct URL are able to access.

Figure 14: If the URL is correct, the teacher's 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.

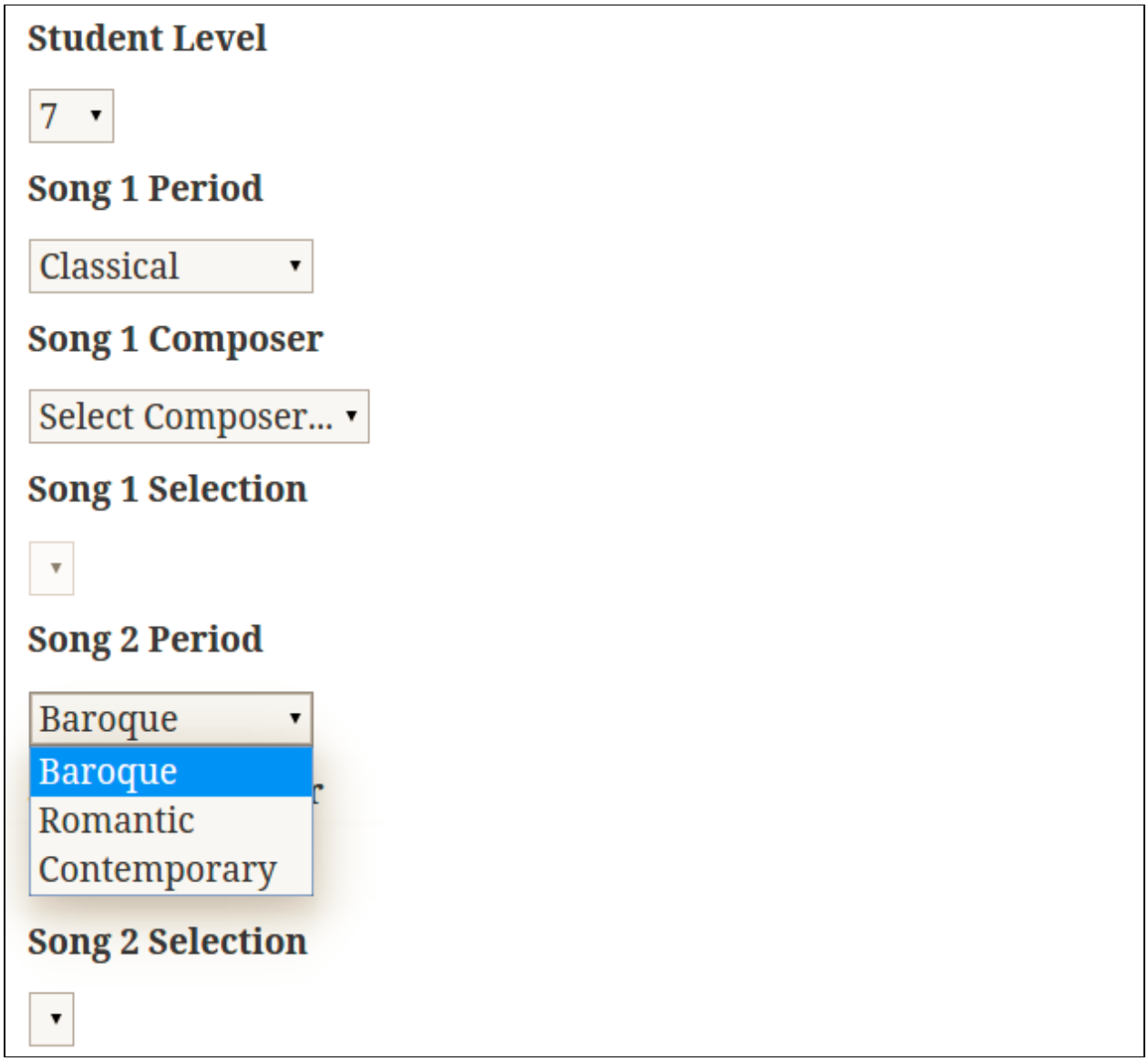


Figure 15: Teachers are responsible for selecting the two songs which their student will be performing. Since they must be from two different periods, if the first song is from the Classical period, then this is removed as an option for the second song's period. This eliminates the aspect of human error of selecting two songs from the same period, which is not allowed.

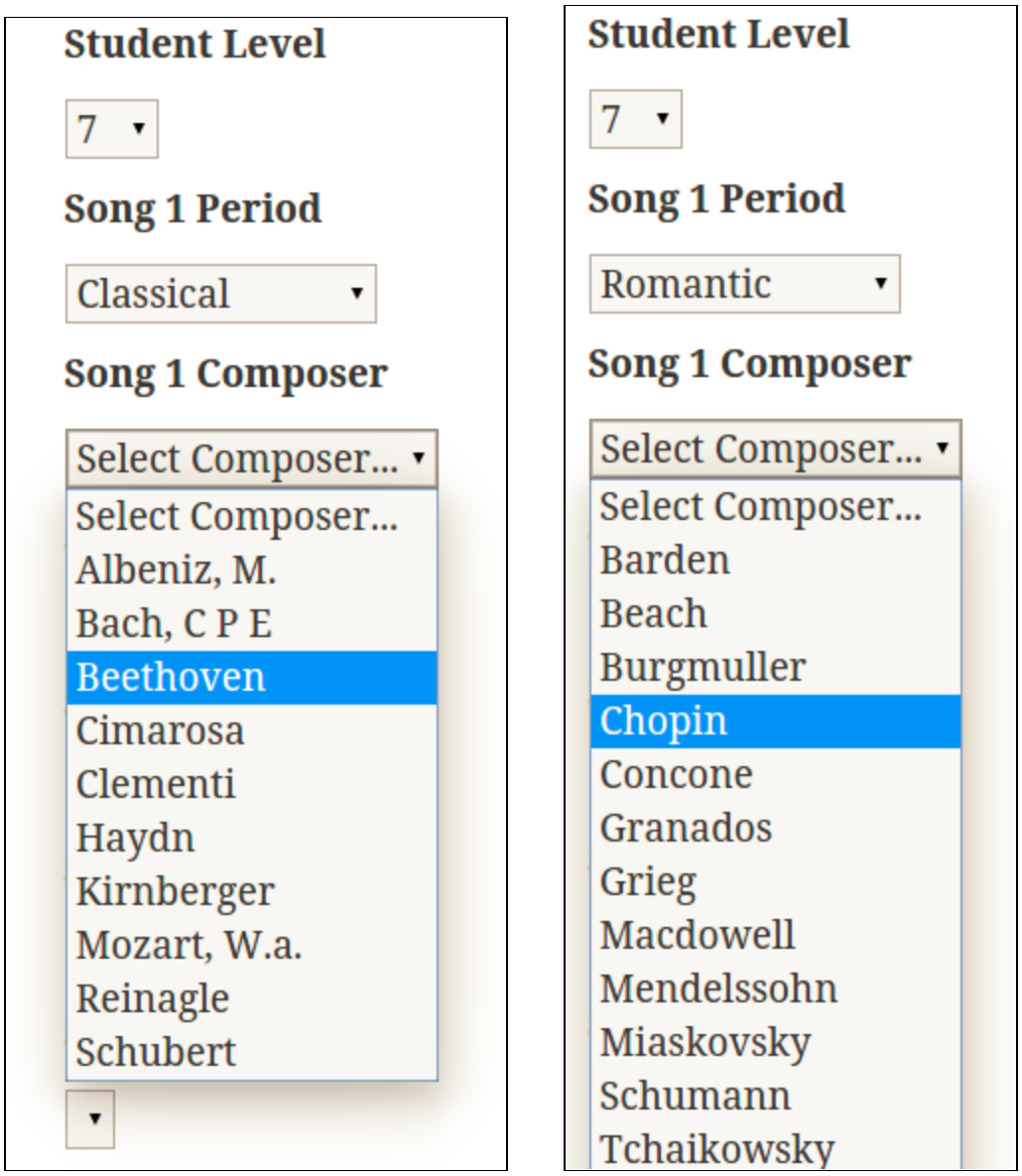


Figure 16.1 and 16.2: Once the song's period is chosen, a list of composers from that period are dynamically populated. If the teacher changes the period, the list of composers is repopulated with the composers from the new period. The composers displayed are the composers with only songs approved for the student's level.

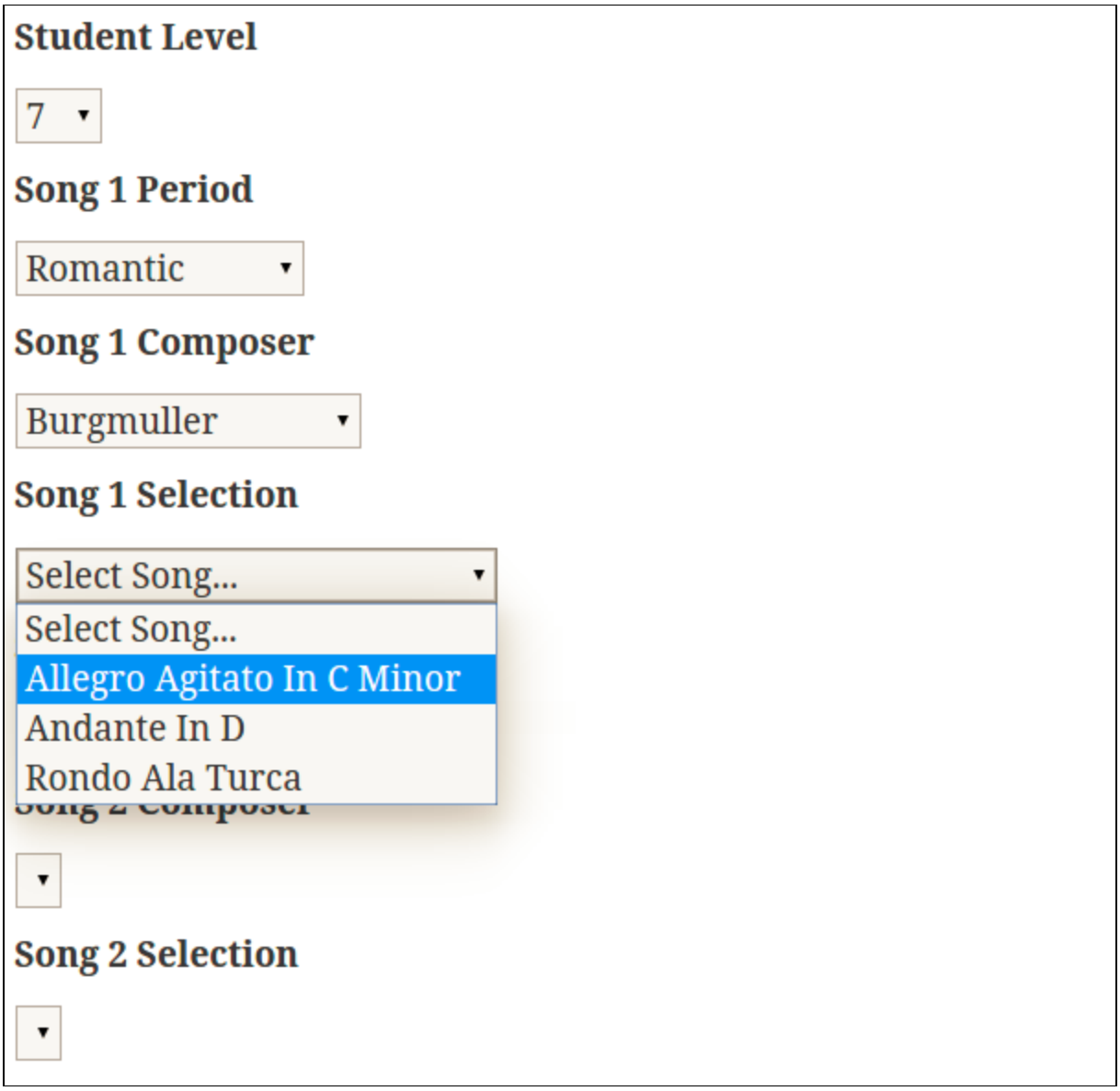


Figure 17: Once the song's composer is chosen, the list of songs for that composer are populated. These songs are specific to the student's level, so there will be no error of a student performing a song which is not approved for their level. If the composer is changed, then the list of song options is repopulated with songs for the new composer.

Glossary of Terms (New Terms)

- Chairman
    - The person in charge of scheduling and managing festivals and music competitions.
- Master Class
    - A class given by an expert of a particular discipline to a student of that discipline.
- Opus Number
    - A separate composition or set of compositions by a particular composer, usually ordered by date of publication.
- Proctor
    - The person who manages each section during the competition. This person would make sure the competition runs smoothly as possible.
- Theory Score
    - A score that the students receive after taking a theory exam which is required for festivals.

References

- Problem-domain book
    - Professional WordPress Plugin Development
      Brad Williams - Ozh Richard - Justin Tadlock - Wiley Pub. - 2011
    - This book shows the methods used in wordpress in order to create self-hosted-blogs and sites, as well as the components used to make the website function. This book has advanced plugin development guides and gives documentation for professional practices used in wordpress development. The book teaches how to utilize hooks, store settings, create translation files, secure plugins, set user roles, integrate widgets, work with JavaScript and AJAX, and create custom posts. The book is written in a practical way in order to lead the reader to development.
- Journal papers, conference papers, and technical reports
    - A Growth Model for Form Generation
        - Rosenman, M. A., "A Growth Model for Form Generation Using a Hierarchical Evolutionary Approach." (1996). Computer-Aided Civil and Infrastructure Engineering, 11: 163–174. doi: 10.1111/j.1467-8667.1996.tb00320.x
        - This paper discusses the approach and model one can use to design. The paper argues that the evolutionary approach to design can be used in conjunction with the generate and test approach. The paper

argues that the evolutionary approach is good for when there are relationships between complex design elements which may be dynamic. This paper shows how the evolutionary approach can be used for synthesis and evaluation during the design process. These claims are shown throughout this paper through the context of designing a house.

- Awards, Success and Aesthetic Quality in the Arts
  - Ginsburgh, Victor. "Awards, Success and Aesthetic Quality in the Arts" (2003). The Journal of Economic Perspectives, Volume 17, Number 2.
  - In this paper, Victor Ginsburgh analyzes the effect that expert opinion on a subject matter has on the success of a product. This is done by analyzing the success of works using awards such as the Oscars, Golden Globes, and the Booker Prize and their economic impacts as examples. Furthermore, Ginsburgh analyzes the effect of rankings of pianists in the The Queen Elisabeth Piano Competition and on their presence in catalogs. Ginsburgh concludes that there is a correlation between awards and success and also that awards are bad predictors of fundamental quality or talent.
- Automated creation of a forms-based database query interface
  - Jayapandian, Magesh and Jagadish, H. V. "Automated creation of a forms-based database query interface" (2008) Proc. VLDB Endow. 1, 1 (August 2008), 695-709.
  - This paper looks at the design of a forms-based interface. An interface is only able to express a limited set of possible queries on a database. An interface which can represent all possible queries to a database is the ideal interface for a database. This paper looks to maximize an interface's ability to represent the needs of a user while being bound by the number of forms used and the complexity of the forms. Given the schema of a database, this paper creates an automated method for generating forms for the database.
- Expressive query specification through form customization
  - Jayapandian, Magesh and Jagadishm H. V. "Expressive query specification through form customization" (2008). In Proceedings of the 11th international conference on Extending database technology: Advances in database technology (EDBT '08). ACM, New York, NY, USA, 416-427.
  - This paper discusses the form as an easy-to-use interface for users to use to query a database. The paper seeks to solve the fact that in

general forms can only express a limited amount of queries since the set of the queries expressed by the form is bound by expertise and vision of the developer at the time the form was created. This paper proposes a method for users to modify a form to express a missing query. This is done by using a form to generate an expression language that creates the queries.

- Consistency in piano performance evaluation
  - Wapnick, J., Flowers, P., Alegant, M., & Jasinskas, L. "Consistency in piano performance evaluation" (1993). Journal of Research in Music Education, 41(4), 282-292.
  - This paper shows the analysis of an experiment performed where Eighty pianists each listened to 21 trials of solo piano music. Trials consisted of two different performances of the same excerpt, and the same music was played on all trials for any given subject. The subjects were to specify which of the two trials they preferred. The results of the test showed that the use of a number scoring system did not improve the consistency of the subjects' preferences. Subjects who used a rating scale were more consistent when using a non musical scale.
- Marshall University Sonatina Festival & Competition
  - Alves, Júlio Ribeiro, "Marshall University Music Department Presents the Marshall University Sonatina Festival & Competition" (2011). *All Performances*. Book 426. http://mds.marshall.edu/music_perf/426
  - This document shows an appropriate example of a music schedule that is produced. The document is of a music festival and competition presented at the Marshall University Sonatina. The document shows the time a piece was played, the piece that was played, the performer, and the adjudication of the performance. The overall schedule of events was shown with descriptions of each event that was happening throughout the event. The biography of both of the adjudicators of this event was also included.
- System and Method for Making Staff Schedules as a Function of Available Resources as Well as Employee Skill Level, Availability and Priority
  - Randall K. Fields, Paul R. Quinn, Todd Blackley, "System and Method for Making Staff Schedules as a Function of Available Resources as Well as Employee Skill Level, Availability and Priority"

        - This patent seeks to generate an optimal staff schedule based on available resources, employee skill level, availability, and priority. This optimal staff schedule is generated for personnel at a remote location by applying a central location policy to a remote location. This is done using a database, policy, labor requirements, tasks to be performed, skills for each task, resources needed, and employees with start and stop time. Upon request to create a schedule, the system and method selects the tasks to be performed and generates an optimized display of the required schedules.
    - Demographic and Sponsorship Considerations for Jazz and Classical Music Festivals
        - Steve Oakes, "Demographic and Sponsorship Considerations for Jazz and Classical Music Festivals", http://www.tandfonline.com/doi/abs/10.1080/714005121
        - This article shows that awareness of demographic information of the audience of a music festival is needs to be increased and enhanced. This is done by studying classical and jazz festivals held in the UK and comparing their audiences. The appropriateness of a strategic fit between the demographics of music festival patrons , sponsoring organizations target segments. This information may show that there can be an increased demand for cross-selling of live entertainment services. Furthermore, the factors that impact the recall of festival sponsors are studied.

Contributions of Team Members

Team Member	Time (Hours)	Activity
Renee Iinuma	8	● User Interface Design ● High Level Behavioral Diagram ● Detailed Design ● Program units
Wesley Kepke	8	● Abstract ● Introduction ● High Level Structural Diagram ● Design Classes ● Data Design ● Program Units
Ernest Landrito	8	● Program Units ● Detailed Design ● References
Kyle Lee	8	● Detailed Design ● Glossary of Terms ● Data Design