ARIA 2.0: Continued Development on 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

Emily Huang
David A. Mar Jr.
Vladislav B. Savranschi

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

May, 2017
We recommend that the thesis prepared under our supervision by

David A. Mar Jr.
Vladislav B. Savranschi

entitled

ARIA 2.0: Continued Development on 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, 2017
Abstract

Administration, Registration, and Information Assistant (ARIA) is a plugin that runs on the WordPress platform. The purpose of ARIA is to manage event creation, student and teacher registration, event scheduling, and document generation for the Northern Nevada Music Teacher Association’s (NNMTA) music festivals. In the past, NNMTA managed the registrations and payments for these events through pen-and-paper methods. However, with the number of registered students increasing every year, it is becoming more difficult to track and secure hard-copy records. ARIA 2.0 is a significant upgrade built upon the previously proposed solution to improve the efficiency, reliability, and security of the event management process.
Acknowledgement

The team would like to thank Dr. Frederick Harris (CSE) and Mrs. Cindy Harris (NNMTA) for guiding and advising the team in this project. Additionally, the team would like to thank the former developers, Wesley Kepke and Renee Inuma, for helping us get started on ARIA and supporting us as we continue the development.
Table of Contents

Abstract i

Acknowledgement ii

Table of Contents iii

List of Tables viii

List of Figures ix

List of Illustrations x

Introduction 1

1 Project Concept 2

1.1 Overview 2

1.2 Significance 2

1.3 Languages and Tools 3

1.4 Market Potential 3

1.4.1 Market Analysis 3

1.4.2 Competitive Analysis 4

1.4.3 Competitive Advantage 4

2 Requirements 5

2.1 High Level Business Requirements 5

2.2 Functional Requirements 5

2.2.1 Priority Level = 1 5
2.2.2 Priority Level = 2

2.2.3 Priority Level = 3

2.3 Non-functional Requirements

2.4 User Stories

2.4.1 Registration

2.4.2 Scheduling

2.4.3 Document Generation

2.3 Use Case Modeling

2.3.1 Use Case Diagram

2.3.2 Detailed Use Cases

2.3.3 Detailed Templates

2.4 Requirement traceability matrices

2.4.1 Matrix for Priority Level = 1

2.4.2 Matrix for Priority Level = 2

2.4.3 Matrix for Priority Level = 3

3 Design and Implementation

3.1 Design Diagrams

3.1.1 Event Creation

3.1.2 Student Registration
3.1.3 Event Scheduling
3.1.4 Document Generation

3.2 System Design
3.2.1 System Context Model
3.2.2 Class Diagrams
  3.2.2.1 Event Creation and Registration
  3.2.2.2 Event Scheduling
  3.2.2.3 Event Document Creation
3.2.3 Class Descriptions
3.2.4 Database Tables

3.3 User Interface Design
3.3.1 Portals
3.3.2 Creating an Event
3.3.3 Scheduling an Event
3.3.4 Upload Music
3.3.5 Registration
3.3.6 Modify Schedule
3.3.7 Input Scores
3.4 Documents Design

3.4.1 RTF Documents

3.4.2 CSV Documents

4 Testing

4.1 Acceptance Criteria

4.1.1 Registration

4.1.2 Scheduling

4.1.3 Document Generation

4.2 Testing Workflow

4.2.1 Registration

4.2.1.1 Happy Path Workflow - Admin Portal

4.2.1.2 Happy Path Workflow - Teacher Registration

4.2.1.3 Negative Testing Workflow - Teacher Reg. with Invalid Fields

4.2.2 Scheduling

4.2.2.1 Happy Path Workflow - Create a Schedule

4.2.2.2 Happy Path Workflow - View/Modify a Schedule

4.2.2.3 Negative Testing Workflow - Create a Schedule

4.2.3 Document Generation

4.2.3.1 Happy Path Workflow - Generate Documents
4.2.3.2 Negative Testing Workflow - Generate Doc. without Schedule

5 Future Work

Conclusion

Bibliography

Appendix

A - Glossary
List of Tables

(22) **Table 1**: An overview of generating a festival schedule.
(23) **Table 2**: An overview of generating a trophy list.
(23) **Table 3**: An overview of displaying a teacher portal.
(25) **Table 4**: Traceability matrix for functional requirements priority level 1.
(26) **Table 5**: Traceability matrix for functional requirements priority level 2.
(27) **Table 6**: Traceability matrix for functional requirements priority level 3.
(36) **Table 7**: Class and functions description for Create_Event.
(36) **Table 8**: Class and functions description for Create_Master_Forms.
(37) **Table 9**: Class and functions description for Music_Upload.
(38) **Table 10**: Class and functions description for Teacher_Upload.
(38) **Table 11**: Class and functions description for Registration_Handler.
(39) **Table 12**: Class and functions description for Scheduler.
(39) **Table 13**: Class and functions description for Event_Information_Handler.
(40) **Table 14**: Class and functions description for Day.
(40) **Table 15**: Class and functions description for Time_Block.
(41) **Table 16**: Class and functions description for Section.
(41) **Table 17**: Class and functions description for Student.
(41) **Table 18**: Class and functions description for Documents.
(43) **Table 19**: Class and functions description for Scores.
(43) **Table 20**: Columns for the event table.
(43) **Table 21**: Columns for the student master form table.
(44) **Table 22**: Columns for the teacher master form table.
(44) **Table 23**: Columns for the schedule table.
(44) **Table 24**: Columns for the music table.
(44) **Table 25**: Columns for the scores table.
(45) **Table 26**: Columns for the teacher hash table.
List of Figures

(18) Figure 1: Use cases describing the main features of ARIA and its relation to the users.

(28) Figure 2: Flowchart for event creation.

(29) Figure 3: Flowchart for student registration and payment.

(30) Figure 4: Sequence diagram for event scheduling. The scheduler collects data from the form and the database. From there, it must determine whether or not a schedule can be created and, as a result, either produce an error or produce a schedule.

(31) Figure 5: Sequence diagram document generation. The documents collects and formats information on the scheduler and student scores.

(32) Figure 6: Updated context model for ARIA.

(33) Figure 7: Class diagram showing all of the classes related to event creation and registration. The Create_Event class is the point of origin for the event forms. Upon successful creation of an event, the class triggers the generation of the student and teacher registration forms.

(34) Figure 8: Class diagram for scheduling an event. The Scheduler object generates the schedule using other objects to retrieve and manage the data.

(35) Figure 9: Class diagram for document generation. The Documents class generates various documentation from scheduled competitions.
List of Illustrations

(46) Illustration 1: The GUI of the Admin Web Portal showing the potential administrator functions and a list of active events.
(47) Illustration 2: The GUI of the Teacher Web Portal that lists contact information and a list of the teacher’s currently registered students.
(47) Illustration 3: The GUI of the Documents Web Portal, allowing the user to configure which of the files are to be generated.
(48) Illustration 4: A short section of the Create an Event form showing some of the fields and whether they are required.
(49) Illustration 5: Another short section of the Create an Event form showing date pickers for the Start/End dates and modified text fields courtesy of the Gravity Forms plugin.
(50) Illustration 6: A short section of the Schedule an Event form showing a drop down list for active events and some required fields.
(51) Illustration 7: The Upload Music page where the administrator will upload their music CSV file.
(52) Illustration 8: A short section of the Student Registration form showing several of the required fields.
(53) Illustration 9: The GUI for the Event Scheduler showing time blocks for Saturday and Sunday. Modifications can be made manually by dragging and dropping students between sections, time blocks, and days.
(54) Illustration 10: Start page UI for judges to input student scores for a certain active event.
(55) Illustration 11: Adjudication form (.rtf), one for each participating student. Alternative text exists for Master Class and Non-Competitive sections.
(56) Illustration 12: Announcing sheet (.rtf), one for each section. Alternative text exists for Master Class and Non-Competitive sections.
(57) Illustration 13: Results sheet (.rtf), one for each section. Alternative text and keys exist for Master Class and Non-Competitive sections.
(58) Illustration 14: Teacher master document (.rtf), one for each teacher. Contains a list of the teacher’s students, together with their performance pieces and session assignments.
(59) Illustration 15: Session Assignments document (.rtf), listing the assigned volunteers for each room within each session.
(59) Illustration 16: Master Schedule document (.csv), listing all scheduler information.
(60) Illustration 17: Simplified Schedule document (.csv), listing basic student information.
(61) Illustration 18: Teacher list document (.csv), listing basic teacher information.
(61) Illustration 19: Theory scores document (.csv) column headers. Students not shown.
Introduction

Administration, Registration, and Information Assistant (ARIA) is a WordPress plugin which was originally developed last year by Team KREW (Kyle Lee, Renee Iinuma, Ernest Landrito, and Wesley Kepke) and has been further developed by Team DEV (David Mar, Emily Huang, Vladislav Savranschi). Sponsored by the Northern Nevada Music Teachers Association (NNMTA), a music organization in Reno, Nevada, the software assists in running the Reno Youth Music Festival[1].

In this popular event, music students of various levels perform in front of judges and afterwards receive feedback and ratings. The music festival occurs twice in the spring: one for the upper level and one for the lower level. Each event consists of registering the students and teachers for the event, scheduling the students, and generating the necessary documents (e.g. performance times, volunteer roles) for the day of the event.

Handling the paperwork manually had been a time-consuming process with many room for potential human errors. The process became even more difficult as the number of registered students increased annually. By utilizing ARIA, the organization can spend less time setting up the music festival by letting the plugin manage the forms.

The core features of ARIA 2.0 were implemented and tested, before the project was used for the 2017 Upper Festival division of the Reno Youth Music Festival in February, and used again for the 2017 Lower Festival division which is going live in May 2017 with fine-tuned and further updated features.
1 Project Concept

1.1 Overview

The features that have been implemented by the original team involved creating the music festival event and associated forms, receiving registration information and payments, and scheduling performance times for students. These functions were in need of heavy modifications, bug fixes, and optimizations to suit the customer requirements. The scheduling and document generation functionality were almost completely redesigned to include new features that cater towards client’s feedback.

Along with maintaining code from last year, the team goals involved designing and implementing new functionality. This includes the creation of an administrator portal, teacher portal, and document generation portal as new web pages to allow easier access to students, teachers, and scheduler configurations without having to browse the WordPress Dashboard directly.

The main users of this product are the members of NNMTA that are running the festival, the students and parents that are entering or attending the festival, and the students’ music teachers. This plugin will primarily focus on the NNMTA Reno Youth Music Festival. ARIA allows for NNMTA to streamline the complicated process of running and managing the festival without having to do the work by hand as it was done in the past.

1.2 Significance

All users will benefit from the plugin as it speeds up the time spent on processing information and managing the event. Instead of manually handling all the paperwork, the organizers can spend a couple minutes setting up information on the music event, and the
plugin can take care of everything else. This project would benefit teachers as they would have an easier time monitoring the students and making sure that the registration forms are filled correctly. The students and parents would benefit from the project as they can perform all the necessary registration and transactions online.

1.3 Languages and Tools

The platform ARIA is built upon is called WordPress, a toolkit used for easy web development for a wide variety of applications[2]. Gravity Forms is a licensed tool that enables ARIA to create forms, and handles all the SQL-querying needed for databasing information automatically[3]. PHP is the language that builds the back-end logic of the plugin, and JavaScript is what is used to create the front-end client and user interface. As such, the project will improve programming, web development, and team working skills of all members involved.

1.4 Market Potential

1.4.1 Market Analysis

This project was requested by NNMTA, and many of its features are specifically catered to serve this group. Therefore, if the project were to be marketed as is, it would have very limited commercial potential. However, ARIA could be updated to have a wider scope so it can accommodate other student-teacher music groups. This would include refactoring many of the entry forms to be more generalizable and removing any NNMTA branding. ARIA could be modified further so that it can be used by any group that seeks to organize event registrations through WordPress.
1.4.2 Competitive Analysis

A brief search reveals that there are many registration management systems available for purchase. For example, RegPack advertises itself as an online registration application that can accept online payments, manage users, send emails, and perform various CMS-like automations[6]. Pricing starts at $49 per month for one administrator account and scales up to $249 per month. Another, RegFox, covers similar features and also integrates with WordPress[7]. RegFox does not set a monthly price; instead they take 99 cents for every registrant. There are even larger applications such as RegOnline which handle event registrations, user management, and even hotel and travel[8]. As is, all of the above applications should be considered direct competitors of ARIA since they possess the same features or include additional functionality.

1.4.3 Competitive Advantage

This project’s main competitive advantage is that it serves a very specific need for a very specific community. ARIA’s registration forms and event management system cater specifically to NNMTA’s events and workflow. ARIA also has an automatic scheduling system that can place students in open slots. None of the competitor products openly advertised such a capability. In addition, this project is being delivered to NNMTA free-of-charge. Compared to the available commercial solutions, ARIA is custom-built, lightweight, and budget-friendly.
2 Requirements

2.1 High Level Business Requirements

1. The ARIA plugin should provide an online electronic system that can manage competition registrations, payments, and scheduling.

2. There should be web pages that allow administrators to create, schedule, and modify a competition.

3. There should be a document generation page to obtain schedule information needed for the day of the festival.

4. There should be web pages to allow users to register students and submit payment for their registrations.

5. There should be web portals that teachers and administrators can access to manage competition and student details.

2.2 Functional Requirements

2.2.1 Priority Level = 1

1. The system shall provide a form to create a competition. The required fields are as follows:

   a. Festival Chairman Email
   b. Competition Name
   c. Competition Start and End Dates
   d. Competition Locations (Street Address, City, State, ZIP, Country)
   e. Student Registration Start and End Dates
   f. Teacher Registration Start and End Dates
g. Volunteer Options for Teachers
h. Volunteer Time Options for Teachers
i. Teacher CSV File (contains teacher names, phone numbers, and email)
j. Command Performance Time Options for Students
k. Master Class Sections
l. Notification Options for Student Registrations (Yes/No)
m. PayPal Account Email Address
n. Pricings for Student Levels (Levels 1-11)

2. After creation of a competition, public pages and links shall be generated for Student and Teacher Registration respectively.

3. The system shall provide a Music Upload function where a CSV file can be uploaded and entries will be automatically added to the database.

4. The system shall use the music database to provide song selection options in the teacher registration form.

5. The system shall provide a form to register a student. The required fields are as follows:
   a. Parent Name (first and last)
   b. Parent Email Address
   c. Student Name (first and last)
   d. Student Birthday
   e. Teacher Name
   f. Available Festival Days
   g. Preferred Command Performance Time
6. After student registration, the system shall redirect the user to PayPal to complete registration payment.

7. After payment, the system shall redirect the user back to the ARIA webpage.

8. After student registration, the system shall send an email to the parent’s email address confirming their child’s registration.

9. After a verified PayPal transaction, the system shall send an email to the selected teacher that their student has registered.

10. The system shall generate a unique hashed link that the teacher can access and complete their student’s registration.

11. The system shall provide a form for teacher registration. The required fields are as follows:

   a. Teacher Name (first and last)
   b. Email Address
   c. Phone Number
   d. Scheduled to Judge for the Festival (Yes/No)
   e. Student Name (first and last)
   f. Student Level
   g. Song 1 Period
   h. Song 1 Composer
   i. Song 1 Selection
   j. Song 2 Period

   h. Student Level
   i. Agree to Compliance Statement
k. Song 2 Composer

l. Song 2 Selection

m. Level 11 Registration Fields (should only be completed if the student is Level 11)
   i. Song 2 Composer
   ii. Song 2 Piece Title

n. Combined Timing of Pieces (minutes)

o. Student Division (Traditional/Non-Competitive/Master Class)

p. Theory Score

q. Check if alternate theory exam was completed

12. The system shall provide a form to schedule the music festival event. The required fields are as follows:

a. Active Competitions

b. Length of Time Blocks (minutes)

c. Number of Time Blocks on Saturday and Sunday

d. Time Block Starting Times for Saturday and Sunday

e. Number of Concurrent Sections on Saturday and Sunday

f. Total Number of Master Class Sections on Saturday and Sunday

g. Amount of Times a Song Can Appear in a Section

h. Master Class Adjudicator Instruction Time

i. Saturday Room Names/Numbers

j. Sunday Room Names/Numbers

k. Saturday Competition Location
1. Sunday Competition Location

m. First Day of Competition

n. Second Day of Competition

13. The system shall provide a form to schedule the command performance. The required fields are as follows:

a. Active Events

b. Command Performance Location (prefilled but editable)

c. Command Performance Date

d. Number of Sessions

e. Session Duration

f. Start Times

g. Room

14. The system shall generate a competition schedule based on provided input. If a schedule cannot be created, then an appropriate warning will be issued.

15. After competition schedule generation, the system shall provide a UI for the user to modify placement of students and sections via drag and drop.

16. The system shall provide an option to add additional teachers that were not included in the CSV file used in competition creation.

17. The system shall provide an option to modify an existing scheduled competition.

18. The system shall provide a way for judges to input scores for student performances.

19. The system shall provide an option to resend a teacher registration link.

20. The system shall provide a web portal for teachers where they can see a list of their students and their contact information.
21. The system shall provide a private web portal for administrators where they can view and access the following:

   a. Create a Competition
   b. Schedule a Competition
   c. View/Modify a Schedule
   d. Upload Music
   e. Add a Teacher
   f. Input Student Scores
   g. Resend Teacher Registration Link
   h. Documents
   i. List of all created competitions
   j. Event CRUD page
   k. Student CRUD page
   l. Teacher CRUD page
   m. Student Schedules
   n. Schedule Command Performance
   o. View/Modify Command Performance

22. The system shall provide a method to publish a schedule to teacher and parents via email.

23. The system shall provide a way to schedule the Command Performance.

24. The system shall provide a way to publish the Command Performance.
2.2.2 Priority Level = 2

1. The system shall provide options to generate the following documents:
   a. Adjudication sheet
   b. Ratings sheet
   c. Competition overview
   d. Competition statistics
   e. Student certificates
   f. List of students, scores, and performance order
   g. Master schedule
   h. Simplified schedule
   i. Room Schedule
   j. Announcement sheets
   k. Teacher list
   l. List of teachers and their contact information
   m. Labels for session file folders and teacher envelopes
   n. Trophy list
   o. Command performance program text
   p. Repertoire list for judges
   q. Volunteer information list
   r. Teacher availability chart

2. The system shall generate these documents in a human-readable format and they should be available for download by the administrator.
3. The system shall support multiple concurrent competitions (competitions that are scheduled on the same days and time slots).

4. The system shall handle music uploads in a timely and efficient manner with no timeouts.

5. The system shall provide a page that the administrator can access view and assign volunteer tasks to teachers.

6. The system shall schedule students based on age, level, and preferences.

7. The system shall update schedule information based on updates made on student and/or teacher information.

8. The system shall contain a Document generation portal for selecting documents to download.

2.2.3 Priority Level = 3

1. The system shall incorporate Google Maps APIs to provide a map to display competition locations.

2. The system shall have web pages that are responsive and viewable on the most common mobile platforms.

3. The system shall accept multiple payment methods besides PayPal.

2.3 Non-functional Requirements

1. The system shall operate on the Wordpress platform as an extensible plugin that can be activated and deactivated.

2. The system shall utilize PHP for its backend page hooks and handling.
3. The system shall utilize the Gravity Forms Wordpress plugin to handle form generation and database records.

4. The system shall utilize the Gravity Forms Web API to retrieve form information to display on the web pages.

5. The system shall implement form verification and notify users if required fields are incomplete or inaccurate (e.g. end dates occurring before start dates).

6. The web pages should be viewable and functional on all current modern web browsers (Chrome, Firefox, Edge, Safari).

7. The web pages should follow all current PHP, HTML, CSS, and Javascript standards.

8. The system shall complete the music syllabus file upload process in less than five minutes.

9. The system shall be able to generate the required documents in less than five minutes.

10. The system shall send emails to parents and teachers within one minute of successful registration.

11. The system shall confirm successful student registration by verifying with PayPal’s Instant Payment Notification (IPN) system.

12. The system shall use Wordpress’s Administrator login credentials to allow access to the private ARIA Administrator web portal and its child pages.
2.4 User Stories

The project is divided into three main subsystems: registration, scheduling, and document generation.

2.4.1 Registration

The registration component includes the web portals and the student/teacher registration pages. The design, implementation, and maintenance of this subsystem is the responsibility of Emily Huang.

The student registration page is accessible to the public and this is where parents can enter their child’s information to enroll for an event. The teacher registration page can only be accessed by a teacher’s unique hash identifier. These links are sent to their respective teachers when a student is registered.

There are two portals, one for the event administrator and one accessible by teachers. The admin portal provides the administrator an up-to-date view of current scheduled events, teachers, and registered students. Administrators can even create, update, and delete event data through this page. The teacher portal lists event and student information. From here, teachers can manage their students’ registrations.

1. As an event administrator, I need a portal that will display event, teacher, and student information.

2. As an event administrator, I need to be able to create, update, and delete event data from the admin portal.
3. As an event administrator, I need to be able to see how many students have currently registered for an event and how many are awaiting teacher completion.

4. As an event teacher, I need a portal that will display a list of my students who have registered for the event.

5. As an event teacher, I need to be able to navigate to the teacher registration for a student from the teacher portal.

2.4.2 Scheduling

The scheduling component contains a form for the music festival and a form for the command performance. After submitting the form, the subsystem will output a schedule containing information from the form and student information. David Mar is responsible for this subsystem.

1. As an event administrator, I need a schedule form so that I can create a schedule containing ample slots for scheduling students.

2. As an event administrator, I need a table displaying information on x days, y time blocks, and z sections because I want a visual of the current schedule to view and easily spot errors.

3. As an event administrator, I need a workable schedule interface because I may want to edit information on location, performance time, and student assignment.

4. As an event administrator, I want to be able to save a modified schedule so that I will not have to make the same modifications when viewing the same schedule.
5. As an event administrator, I want to email parents and teachers the schedule information (that is relative to the person) so that they know when their child/student will perform.

2.4.3 Document Generation

The document generation component can be accessed by the user through the button “Generate Documents” found within the form following the scheduling process, or during the modification of an existing schedule. After the button is pressed, the subsystem will pull all scheduler information from the database, generate the files using PHP RTF-Lite, package each file into a single compressed “.zip” folder, and output the package containing the requested information to be downloaded by the user. Vladislav Savranschi is responsible for this subsystem.

1. As an event administrator, I need to be able to generate Adjudication forms, detailing the number of the anonymous performer, number of the anonymous teacher, the performer’s level, a list of the pieces (with their respective composers) which will be performed. The judges must be allowed to fill out the sheet by selecting one of the pieces, must be able to provide a rating, and general comments on the performance. The Adjudication form must also contain event name, date, time, and room number of the performance. This document will be delivered as a text document (Rich Text File).

2. As an event administrator, I need to be able to generate Result sheets, detailing each student’s name, a list of the pieces (with their respective composers), and a list of the possible scores that the students can receive. The judges must be able to select
which piece was presented, and which score was received by that student. The Results sheet must also contain event name, date, time, and room number of the performance. This document will be delivered as a text document (Rich Text File).

3. As an event administrator, I need to be able to generate the Master schedule, detailing a list of sections (including date, time, room, format, and room monitor/volunteer). Each section will contain a column for the teachers, and a column with their corresponding students. There must be a column for the student’s age, level, and the pieces (with their respective composers). This document will be delivered as a spreadsheet (Excel).

4. As an event administrator, I need to be able to generate the Simplified Master schedule, detailing a list of sections (including date, time, room, format). Each section will contain a column for the students. There must be a column for the student’s level. This document will be delivered as a spreadsheet (Excel).

5. As an event administrator, I need to be able to generate the documents to appear distinct for each student that is part of the Masterclass format, or the Command performance. On the Adjudication form, the rating must instead appear as a checkbox for whether the student is approved to go on to the command performance. On the Result sheet, the possible ratings for Traditional (Superior with Distinction, Superior, Excellent, Needs Attention, Non-Competitive, Withdrawn) must be modified to an alternative set of ratings (Command Performance, Masterclass but not Command Performance, Non-Competitive, Withdrawn).
2.3 Use Case Modeling

2.3.1 Use Case Diagram

Figure 1: Use cases describing the main features of ARIA and its relation to the users.
2.3.2 Detailed Use Cases

1. CreateCompetition (UC_01)
   The chairman can create an event after providing information about that event, such as event name, event dates, event location, and event fees. Necessary registration forms and databases will be created for the event.

2. UploadMusicInfo (UC_02)
   The chairman can upload a list of music that can be played in the event. The list to be uploaded must follow a specific format; otherwise, an error will occur.

3. DisplayChairmanPortal (UC_03)
   This will provide the chairman controls for the event. Some controls include viewing student and teacher information, generating schedules and documents, and modifying and information.

4. ModifyStudentInformation (UC_04)
   The chairman can change any information regarding a student that is registered for a given event.

5. ModifyTeacherInformation (UC_05)
   The chairman can change any information regarding a teacher that has students or is judging in a given event.

6. GenerateEventSchedule (UC_06)
This provides a basic structure to generate a schedule for an event. Once the type of event and event options have been determined, it will collect the necessary data and generate a schedule.

7. ModifyEventSchedule (UC_07)

Once a schedule has been created, the chairman can make any changes to the schedule. The chairman and then either save the changes and update the schedule or cancel the changes and keep the original schedule.

8. GenerateFestivalSchedule (UC_08)

This will generate a schedule specifically for the Reno Youth Music Festival. The schedule will include student information, student songs, and performance time and location.

9. GenerateEventDocuments (UC_09)

This will create documents based on the schedule and student and teachers information and display them. This will be for the day of the event and allow participants to know their destinations.

10. InputJudgeScore (UC_10)

This will be done after the student has performed and the judges have written their score. The chairman will input scores to a form. Once completed, data from the forms will be extracted and stored in the database.

11. GenerateTrophyList (UC_11)
The chairman can create a list of students who receive an award based on their scores. Only the students who received a Superior, a Superior with Distinction, or Command Performance will be listed.

12. GenerateCommandPerformanceSchedule (UC_12)
   This generates a schedule for students who are playing in a command performance. The schedule includes student information, judges’ song of choice, and performance date and time.

13. GenerateEventStatistics (UC_13)
   The chairman can compile data from the whole event, from registrations to judges score to the command performance, and create statistics to analyze the event.

14. RegisterStudent (UC_14)
   After the student has filled out the student registration form, the data will be collected and stored in the student database.

15. PayEventFees (UC_15)
   This students will pay for the event after they filled out their form. This must be done to complete the student portion of the registration. The total cost will depend on the student level.

16. RegisterTeacher (UC_16)
   After the student has filled out the teacher registration form, the data will be collected and stored in the student database and the teacher database.

17. DisplayTeacherPortal (UC_17)
This will create a list of students that has completed their portions of the registration process. If the teacher selects a student, the teacher will then be provided a teacher registration form for that student.

18. DisplayRegisteredStudentsInfo (UC_18)

This collects data from the students under a specified teacher and displays the information on the page.

19. PublishScheduleInfoParent(UC_19)

This emails parents information regarding their child’s scheduled performance.

20. PublishScheduleInfoTeacher(UC_20)

This emails teachers information regarding their student’s scheduled performance.

2.3.3 Detailed Templates

Shown below are templates for three different use cases:

**Table 1**: An overview of generating a festival schedule.

<table>
<thead>
<tr>
<th>Use Case: GenerateFestivalSchedule</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID:</strong> UC_08</td>
</tr>
<tr>
<td><strong>Actor:</strong> Chairman</td>
</tr>
<tr>
<td><strong>Preconditions:</strong></td>
</tr>
<tr>
<td>1. A schedule for the festival has been generated.</td>
</tr>
<tr>
<td><strong>Flow of Events:</strong></td>
</tr>
<tr>
<td>1. This use case happens after the students and teachers have registered for the event.</td>
</tr>
</tbody>
</table>
2. The system obtains information from student database regarding student information, performance day and time, and location.
3. The system generates documents using student information
4. The system obtains information from teacher database regarding judging time and location.
5. The system generates documents using judge information
6. The system obtains information from teacher database regarding volunteer duty and time.
7. The system generates documents using teachers information

Postconditions:
1. Student documents are generated
2. Judging documents are generated
3. Volunteer documents are generated

**Table 2**: An overview of generating a trophy list.

<table>
<thead>
<tr>
<th>Use Case: GenerateTrophyList</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID</strong>: UC_11</td>
</tr>
<tr>
<td><strong>Actor</strong>: Chairman</td>
</tr>
</tbody>
</table>

**Preconditions:**
1. Scores for each student has been inputted into the database.

**Flow of Events:**
1. This use case happens after the day of the festival.
2. Add students who received high enough score from the judges to a new list
3. Generate a document displaying the student name and score

**Postconditions:**
1. The webpage displays a list of students who reached a certain score or higher

**Table 3**: An overview of displaying a teacher portal.

| Use Case: DisplayTeacherPortal |
**ID: UC_17**

**Actor:** Teacher

**Preconditions:**

1. Teacher must have at least one student registered, i.e. in the student database.

**Flow of Events:**

1. The use case starts when the teacher clicks on a link (given through e-mail) leading to his or her teacher portal.
2. The system finds the students belong to the teacher and stores them in a new list.
3. For each student in the newly created list
   - 3.1. Find link to teacher registration form associated with student.
   - 3.2. Display student name.
   - 3.3. Display link on the side of student name.

**Postconditions:**

1. The webpage displays a list of students associated with the teacher.
2. The webpage provides links that leads to teachers registrations page associated with a student.
2.4 Requirement traceability matrices

2.4.1 Matrix for Priority Level = 1

Table 4: Traceability matrix for functional requirements priority level 1.

| Use Cases | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|-----------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1         | X | X | X |   |   | X |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 2         |   |   |   |   |   | X |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 3         | X |   | X | X | X | X |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 4         | X | X | X |   | X |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 5         | X | X |   | X | X | X |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 6         |   | X | X | X | X | X |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 7         |   | X |   | X | X |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 8         |   | X | X | X | X | X |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 9         |   | X |   | X | X |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 10        |   | X |   | X | X |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 11        |   | X |   | X | X | X |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 12        |   | X |   | X | X | X |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 13        |   | X | X | X | X | X |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 14        |   | X | X | X | X | X |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 15        |   | X | X | X | X | X |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16        |   | X | X | X | X |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 17        |   | X |   | X | X |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 18        |   | X | X | X | X |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 19        |   | X | X | X | X |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 20        |   | X | X | X | X |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
2.4.2 Matrix for Priority Level = 2

**Table 5:** Traceability matrix for functional requirements priority level 2.

<table>
<thead>
<tr>
<th>Use Cases</th>
<th>Function Requirements Priority Level = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>X X X</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>X X X</td>
</tr>
<tr>
<td>7</td>
<td>X X</td>
</tr>
<tr>
<td>8</td>
<td>X X X</td>
</tr>
<tr>
<td>9</td>
<td>X X</td>
</tr>
<tr>
<td>10</td>
<td>X X</td>
</tr>
<tr>
<td>11</td>
<td>X X</td>
</tr>
<tr>
<td>12</td>
<td>X X X X</td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>
### 2.4.3 Matrix for Priority Level = 3

**Table 6:** Traceability matrix for functional requirements priority level 3.

<table>
<thead>
<tr>
<th>Use Cases</th>
<th>Priority Level = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>
3 Design and Implementation

3.1 Design Diagrams

The processes for event creation, registration, scheduling, and document generation.

3.1.1 Event Creation

![Flowchart for event creation](image)

Figure 2: Flowchart for event creation.
3.1.2 Student Registration

![Flowchart for student registration and payment.](image)

**Figure 3:** Flowchart for student registration and payment.
3.1.3 Event Scheduling

[Sequence Diagram]

Figure 4: Sequence diagram for event scheduling.
3.1.4 Document Generation

Figure 5: Sequence diagram document generation.
3.2 System Design

3.2.1 System Context Model

The following model shows the primary systems of ARIA: event creation and scheduling, student registration, and document generation. ARIA also relies on a database system which is provided by the Gravity Forms plugin.

Figure 6: Updated context model for ARIA.
3.2.2 Class Diagrams

3.2.2.1 Event Creation and Registration

Figure 7: Class diagram showing all of the classes related to event creation and registration. The Create_Event class is the point of origin for the event forms. Upon successful creation of an event, the class triggers the generation of the student and teacher registration forms.
3.2.2.2 Event Scheduling

Figure 8: Class diagram for scheduling an event. The Scheduler object generates the schedule using other objects to retrieve and manage the data.
3.2.2.3 Event Document Creation

Figure 9: Class diagram for document generation. The Documents class generates various documentation from scheduled competitions.
3.2.3 Class Descriptions

The following tables explain the classes used for event creation and registration, event scheduling, and event document creation.

**Table 7**: Class and functions description for Create_Event.

<table>
<thead>
<tr>
<th>Create_Event</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>This class</strong> defines event creation functionality. It will create the necessary Wordpress forms and pages for the event as well as the student/teacher registration forms.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <code>create_event_activation</code> - starts the process of creating the event form</td>
</tr>
<tr>
<td>- <code>create_teacher_and_student_forms</code> - creates the new student/teacher registration forms for this event</td>
</tr>
<tr>
<td>- <code>create_event_form</code> - creates a new Gravity Form object with all the associated fields for the event</td>
</tr>
<tr>
<td>- <code>create_event_form_validation</code> - called after Create Event form submission and performs form validation</td>
</tr>
<tr>
<td>- <code>create_student_form</code> - creates a new Gravity Form object for student registration</td>
</tr>
<tr>
<td>- <code>student_form_validation</code> - called after student form submission and performs form validation</td>
</tr>
<tr>
<td>- <code>create_teacher_form</code> - creates a new Gravity Form object for the teacher version of student registration</td>
</tr>
<tr>
<td>- <code>teacher_form_validation</code> - called after teacher student form submission and performs form validation</td>
</tr>
<tr>
<td>- <code>unrequire_update_event</code> - a hook that is called when updating an event entry that will set some of the form’s required fields to false (so it will not trigger form validation)</td>
</tr>
<tr>
<td>- <code>presubmission_update_event</code> - a hook that is called that will update the event entry and propagate its changes to related student and teacher forms</td>
</tr>
</tbody>
</table>

**Table 8**: Class and functions description for Create_Master_Forms.

<table>
<thead>
<tr>
<th>Create_Master_Forms</th>
</tr>
</thead>
</table>

---
This class defines the functionality for creating the student and teacher master forms. The master forms are considered sources of truth regarding student and teacher information. These master forms are associated with their respective event.

**Functions**

- **create_student_master_form** - creates the student master form
- **create_teacher_master_form** - creates the teacher master form
- **create_student_master_view_form** - creates the student master view form that is used as a form skeleton for the student CRUD page
- **create_teacher_master_view_form** - creates the teacher master view form that is used as a form skeleton for the teacher CRUD page
- **create_teacher_hash_form** - creates the teacher hash form that is used to associate all teacher hashes with their events, students, and teacher forms
- **presubmission_update_teacher** - a hook that is called to update the teacher master entry from the teacher CRUD page
- **presubmission_update_student** - a hook that is called to update the student master entry from the student CRUD page
- **unrequire_update_student** - a hook that is called to set some of the student required fields to false (to bypass form validation)
- **unrequire_update_teacher** - a hook that is called to set some of the teacher required fields to false (to bypass form validation)

**Table 9: Class and functions description for Music.Upload.**

**Music.Upload**

This class defines the functionality for the music syllabus upload. The music syllabus is uploaded via a specifically formatted CSV file and the entries are saved to the database. The music entries are later used in student registration and event scheduling.

**Functions**

- **create_music_upload_form** - creates the form to upload the music syllabus
- **add_music_from_csv** - opens the submitted CSV file and parses it for music entries. The entries are then added to the database.
- **create_nnmta_music_form** - creates the NNMTA music database where all music entries are saved to
- **remove_all_music_from_database** - clears all music entries from the NNMTA music database
- **music_field_id_array** - returns an associative array with field IDs for the music database
Table 10: Class and functions description for Teacher_Upload.

<table>
<thead>
<tr>
<th>Teacher_Upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>This class defines functionality for uploading a list of teachers. After selecting the appropriate event, a specifically formatted CSV is submitted and then is parsed for teacher entries. These entries are saved to the database to be then used for event scheduling and student registration.</td>
</tr>
</tbody>
</table>

**Functions**
- **create_teacher_upload_form** - creates the teacher upload form
- **before_teacher_upload** - this function pre-populates entries and drop down menus in the teacher upload form
- **after_teacher_upload** - this function calls another method to parse the teacher CSV file and save them to the database.
- **upload_from_csv** - opens the submitted CSV file and parses for teacher entries.
- **teacher_upload_field_id_array** - returns an associative array with field IDs for the teacher database

Table 11: Class and functions description for Registration_Handler.

<table>
<thead>
<tr>
<th>Registration_Handler</th>
</tr>
</thead>
<tbody>
<tr>
<td>This class defines functionality for handling submissions of student/teacher registration for an event. This includes pre-population of registration pages and email notifications that occur after form submission.</td>
</tr>
</tbody>
</table>

**Functions**
- **send_registration_emails** - sends emails to the parent, teacher, and event facilitator of the registered student
- **find_student_entry** - searches through the student master form and checks if the student exists
- **find_teacher_entry** - searches through the teacher master form and checks if the teacher exists
- **check_student_teacher_relationship** - checks if the student and teacher entries are associated. If they are, then the teacher can continue to complete registration for their student.
- **get_teacher_pre_populate** - returns an associative array of teacher information to pre-populate the teacher student registration form
- **get_student_pre_populate** - returns an associative array of student information
to pre-populate the student registration form

- **after_teacher_submission_email** - sends a confirmation email to the teacher that they have successfully completed their student’s registration

<table>
<thead>
<tr>
<th>Table 12: Class and functions description for Scheduler.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scheduler</strong></td>
</tr>
<tr>
<td>This class defines functionality for scheduling students to perform in an event. The schedule for an event can be created or modified based on user-inputted options.</td>
</tr>
<tr>
<td><strong>Functions</strong></td>
</tr>
<tr>
<td>- <strong>retrieve_event_information</strong> - sets an Event_Information_Handler object data pertaining to the event. The data includes event information and student and teacher information.</td>
</tr>
<tr>
<td>- <strong>create_event_schedule</strong> - uses information given to generate a schedule for the event. Returns true if a schedule has been created and false if an error occurred.</td>
</tr>
<tr>
<td>- <strong>is_schedule_possible</strong> - checks if creating a schedule with the provided information is possible. Returns true if it is possible and false otherwise.</td>
</tr>
<tr>
<td>- <strong>get_event_schedule</strong> - returns a string representing data in HTML format. This is used for displaying the schedule on the screen in a table format.</td>
</tr>
<tr>
<td>- <strong>modify_event_schedule</strong> - searches for a schedule of an event and enables the user to modify the schedule.</td>
</tr>
<tr>
<td>- <strong>get_student</strong> - searches for a student within a schedule.</td>
</tr>
<tr>
<td>- <strong>create_schedule_structure</strong> - creates a general schedule consisting a number of days, each of which containing a number of time blocks, each of which containing a number of sections.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 13: Class and functions description for Event_Information_Handler.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Event_Information_Handler</strong></td>
</tr>
<tr>
<td>This class retrieves and holds event data from the database. It is used by the Scheduler class and will only obtain information based on the event ID given.</td>
</tr>
<tr>
<td><strong>Functions</strong></td>
</tr>
<tr>
<td>- <strong>retrieve_event</strong> - retrieves and holds information on the event. The data includes event name, dates, and locations.</td>
</tr>
</tbody>
</table>
• **retrieve_students** - retrieves and holds information on students. The data includes student name, age, teacher, and songs that the student is playing.
• **retrieve_teachers** - retrieves and holds information on teachers. The data includes teacher name, students, and email address.

---

**Table 14:** Class and functions description for Day.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Day</strong></td>
<td>This class sets information regarding students’ participation in the event. It is used by the Scheduler class and describes what day the students will perform.</td>
</tr>
</tbody>
</table>

**Functions**

- **get_date** - returns the date of the student’s performance.
- **get_address** - returns the address of the student’s performance.
- **get_student** - searches through each time block for the student associated with the given student ID. Returns the student if found and NULL if otherwise.
- **schedule_student** - finds a time block suitable to a student’s preferences and assigns student to that time. Returns true if the student has been assigned and false if the student could not be assigned a time

---

**Table 15:** Class and functions description for Time_Block.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time_Block</strong></td>
<td>This class sets information regarding students’ participation in the event. It is used by the Day class and describes what time block the students will perform.</td>
</tr>
</tbody>
</table>

**Functions**

- **get_student** - searches through each section for the student associated with the given student ID. Returns the student if found and NULL if otherwise.
- **schedule_student** - finds a section suitable to a student’s preferences and assigns student to that time. Returns true if the student has been assigned and false if the student could not be assigned a time

---

**Table 16:** Class and functions description for Section.
Section

This class sets information regarding students’ participation in the event. It is used by the Time_Block class and describes what section the students will perform.

**Functions**

- **get_student** - searches through a list of students assigned to the current section for the student associated with the given student ID. Returns the student if found and NULL if otherwise.
- **schedule_student** - finds a section suitable to a student’s preferences and assigns student to that time. Returns true if the student has been assigned and false if the student could not be assigned a time

**Table 17**: Class and functions description for Student.

Student

This class contains information pertaining to the student. The user will be able to add and remove songs that the student plans to perform.

**Functions**

- **add_song** - adds a song that the current student plans to perform. Returns true if the song given is valid, and returns false otherwise.
- **remove_song** - removes a song from the current student. Returns true if the song was removed, and returns false otherwise.
- **get_song** - returns a song that a student plans to perform. The songs are stored in an array, and a song is obtained using an index value. Returns NULL if song is not found.

**Table 18**: Class and functions description for Documents.

Documents

This class defines the functionality for generating various documents used after the competition.

**Functions**

- **generate_documents** - Runs document generator. Entry point into the class.
• **generate_selected_documents** - Compiles and downloads only the documents selected by the configurations set within the documents portal. Downloads all documents if all or none were selected in advance.

• **generate_all_documents** - Compiles and downloads all documents.

• **create_command_performance_program** - Outputs the command performance program (.rtf) document, displaying students together with their performance piece, associated judge and teacher - to be printed for the participants of the command performance event.

• **create_announcing_sheets** - Outputs the program (.rtf) document, displaying students together with their performance piece, associated judge and teacher - to be printed for the participants of the traditional or master class event.

• **create_adjudication_forms** - Outputs the judge forms (.rtf) document displaying the anonymous performer, teacher, together with the performance piece and score box - personalized for each judge for traditional or master class event.

• **create_results_sheets** - Outputs the result sheets (.rtf) document displaying the anonymous performer, the performance pieces, together with a score key and score boxes for tallying results - personalized for each judge for traditional or master class event.

• **create_teacher_to_student_map** - Maps teachers to each of the student data structures for use by the document generator’s algorithms.

• **create_teacher_master** - Outputs the (.rtf) document displaying each teacher’s list of students and their performance pieces - personalized for each teacher for traditional or master class event.

• **create_session_assignments** - Outputs the session assignments (.rtf) document to use as reference for students and teachers looking for their assigned session placements.

• **create_room_schedules** - Outputs the room schedule (.rtf) document to use as reference for students and teachers looking for their assigned room placements.

• **create_master_schedule_csv** - Outputs the complete schedule (.csv) for the event. Each student, together with their associated teacher, judges, volunteers, and their relevant information is displayed within time slots assigned to them by the scheduler.

• **create_simplified_schedule_csv** - Outputs the simplified schedule (.csv) for the event. Each student, together with their associated teacher, judges, volunteers, and a small part of their information is displayed within time slots assigned to them by the scheduler.

• **create_teacher_list_csv** - Outputs the list of teachers (.csv), and their students.

• **create_theory_scores_csv** - Outputs the list of students (.csv), and their theory scores.

• **download_documents** - Packages all generated files into a compressed folder (.zip), prompts the user to accept, and downloads the compressed folder onto the user’s drive.

• **aria_styles** - Style sheet information for formatting the PHPRTFLite library’s
document output style (font sizes, color, spacing, etc).

- **generate_trophy_list** - Outputs a list of students that received specific trophies.

### Table 19: Class and functions description for Scores.

<table>
<thead>
<tr>
<th>Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>This class defines the input and output of scores for students within the database.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>set_student_scores</strong> - sets scores for a student in the database.</td>
</tr>
<tr>
<td>- <strong>get_student_scores</strong> - gets scores for a student in the database.</td>
</tr>
</tbody>
</table>

### 3.2.4 Database Tables

A database is needed in order to store and transfer data from one system to another. Shown below are four tables needed for registration, scheduling, and documenting functionalities.

#### Table 20: Columns for the event table.

<table>
<thead>
<tr>
<th>Event</th>
<th>Facilitator Email</th>
<th>Event Name</th>
<th>Start Date</th>
<th>End Date</th>
<th>Student Reg. Start Date</th>
<th>Student Reg. End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>cont.</td>
<td>Teacher Reg. Start Date</td>
<td>Teacher Reg. End Date</td>
<td>Primary Location Address</td>
<td>Secondary Location Address</td>
<td>Volunteer Options</td>
<td>Volunteer Times</td>
</tr>
<tr>
<td>cont.</td>
<td>Teacher Upload File Path</td>
<td>Paypal Email Address</td>
<td>Teacher Reg. Start Date</td>
<td>Teacher Reg. End Date</td>
<td>Payment Amounts</td>
<td>Scheduled Flag</td>
</tr>
</tbody>
</table>

#### Table 21: Columns for the student master form table.
<table>
<thead>
<tr>
<th>Student</th>
<th>First Name</th>
<th>Last Name</th>
<th>Email</th>
<th>Birthday</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>cont.</td>
<td>Teacher</td>
<td>Songs</td>
<td>Availability</td>
<td>Student Hash</td>
<td>Teacher Hash</td>
</tr>
<tr>
<td>cont.</td>
<td>Payment Status</td>
<td>Registration Status</td>
<td>Teacher Registration URL</td>
<td>Student Public Form Entry ID</td>
<td></td>
</tr>
</tbody>
</table>

**Table 22:** Columns for the teacher master form table.

<table>
<thead>
<tr>
<th>Teacher</th>
<th>First Name</th>
<th>Last Name</th>
<th>Email</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>cont.</td>
<td>Students</td>
<td>Judge</td>
<td>Volunteer Preferences</td>
<td>Email Sent Flag</td>
</tr>
</tbody>
</table>

**Table 23:** Columns for the schedule table.

<table>
<thead>
<tr>
<th>Schedule</th>
<th>ID</th>
<th>Event Name</th>
<th>Number Days</th>
<th>Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>cont.</td>
<td>Students</td>
<td>Judges</td>
<td>Volunteer</td>
<td></td>
</tr>
</tbody>
</table>

**Table 24:** Columns for the music table.

<table>
<thead>
<tr>
<th>Song</th>
<th>Level</th>
<th>Period</th>
<th>Composer</th>
<th>Title</th>
<th>Opus/Catalog Number</th>
</tr>
</thead>
</table>

**Table 25:** Columns for the scores table.

<table>
<thead>
<tr>
<th>Scores</th>
<th>Student</th>
<th>Teacher</th>
<th>Song</th>
</tr>
</thead>
</table>
Table 26: Columns for the teacher hash table.

<table>
<thead>
<tr>
<th>Teacher Hash</th>
<th>Event Name</th>
<th>Hash</th>
<th>Teacher Master Form ID</th>
<th>Teacher Master Entry ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>cont.</td>
<td>Student Master Form ID</td>
<td>Event Entry ID</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.3 User Interface Design

3.3.1 Portals

Illustration 1: The GUI of the Admin Web Portal showing the potential administrator functions and a list of active events.
Illustration 2: The GUI of the Teacher Web Portal that lists contact information and a list of the teacher’s currently registered students.

Illustration 3: The GUI of the Documents Web Portal, allowing the user to configure which of the files are to be generated.
3.3.2 Creating an Event

Illustration 4: A short section of the Create an Event form showing some of the fields and whether they are required.
**Illustration 5**: Another short section of the Create an Event form showing date pickers for the Start/End dates and modified text fields courtesy of the Gravity Forms plugin.
3.3.3 Scheduling an Event

Illustration 6: A short section of the Schedule an Event form showing a drop down list for active events and some required fields.
3.3.4 Upload Music

Illustration 7: The Upload Music page where the administrator will upload their music CSV file.
Illustration 8: A short section of the Student Registration form showing several of the required fields.
3.3.6 Modify Schedule

Illustration 9: The GUI for the Event Scheduler showing time blocks for Saturday and Sunday. Modifications can be made manually by dragging and dropping students between sections, time blocks, and days.
Illustration 10: Start page UI for judges to input student scores for a certain active event.
3.4 Documents Design

3.4.1 RTF Documents

Illustration 11: Adjudication form (.rtf), one for each participating student. Alternative text exists for Master Class and Non-Competitive sections.
Yet Another Test

Announcing Sheet

Saturday, 9 am, Room 1

Traditional

Judge: Judge A, Judge B
Proctor: Cindy Harris
Door Mentor: Fred Harris

Performance Order:

1. Vladislav Savranschi
   English Woods
   Nannerl’s Minuet
   George, Jon
   Mozart, L.

2. David Mar
   By The Seaside
   Bozzini
   Steedshog
   Kreiger

3. Emily Huang
   Bike Ride
   Minuet, F Major
   Rollins
   Mozart, W.A.

4. Bruce Lee
   Minuetto Scherzando
   Innocence
   Seines
   Burgmiller

5. Terry Crews
   March
   Carnival Scene
   Shostakovich
   Corepin

6. Chuck Norris
   By The Seaside
   Bravo, El Toro
   Steedshog
   Noosa

7. Arnold Schwarzenegger
   Distant Bells
   Petits Arabesques
   Steedshog
   Agay

Illustration 12: Announcing sheet (.rtf), one for each section. Alternative text exists for Master Class and Non-Competitive sections.
Yet Another Test

Saturday, 9 am, Room 1

Traditional
Judge: Judge A, Judge B

<table>
<thead>
<tr>
<th>SD: Superior with Distinction</th>
<th>E: Excellent</th>
<th>NC: Non-competitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>S: Superior</td>
<td>NA: Needs Attention</td>
<td>W: Withdrawn</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vladislav Savranschi</th>
<th>SD</th>
<th>S</th>
<th>E</th>
<th>NA</th>
<th>NC</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>George, Jon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mozart, L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>David Mar</th>
<th>SD</th>
<th>S</th>
<th>E</th>
<th>NA</th>
<th>NC</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strabhog</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kreger</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Emily Huang</th>
<th>SD</th>
<th>S</th>
<th>E</th>
<th>NA</th>
<th>NC</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rollin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mozart, W.A.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bruce Lee</th>
<th>SD</th>
<th>S</th>
<th>E</th>
<th>NA</th>
<th>NC</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seixas</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Burgmiller</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Terry Crews</th>
<th>SD</th>
<th>S</th>
<th>E</th>
<th>NA</th>
<th>NC</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shostakovich</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coupelin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chuck Norris</th>
<th>SD</th>
<th>S</th>
<th>E</th>
<th>NA</th>
<th>NC</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strabhog</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noona</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arnold Schwarzenegger</th>
<th>SD</th>
<th>S</th>
<th>E</th>
<th>NA</th>
<th>NC</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strabhog</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agay</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Illustration 13: Results sheet (.rtf), one for each section. Alternative text and keys exist for Master Class and Non-Competitive sections.


Yet Another Test

Cindy Harris’ Student Schedule
The performance schedule for your student(s) is as follows:

Vladislav Savranschi, Saturday, 9 am, Room 2, performer # 4
Vandall
Concone
Queen Anne’s Lace
Etude

David Mar, Sunday, 2 pm, Room 1, performer # 1
Morosky
Alexander
Prelude in E minor
The Village Piper

Emily Huang, Sunday, 2 pm, Room 1, performer # 3
Bach, J.S.
Burgmuller
Minuet, G Major
Ballade

Wesley Kepke, Sunday, 2 pm, Room 1, performer # 4
Maykapar
Gillock
The Butterfly
Deserted Ballroom

Rence Iinuma, Sunday, 2 pm, Room 2, performer # 4
Chaminade
Gurlitti
Aubade
Solero

Illustration 14: Teacher master document (.rtf), one for each teacher. Contains a list of the teacher’s students, together with their performance pieces and session assignments.
Illustration 15: Session Assignments document (.rtf), listing the assigned volunteers for each room within each session.

3.4.2 CSV Documents

<table>
<thead>
<tr>
<th>Section</th>
<th>Format</th>
<th>Lead Student</th>
<th>Age</th>
<th>Level</th>
<th>Piece 1</th>
<th>Piece 2</th>
<th>Judge</th>
<th>Proctor</th>
<th>Door Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saturday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Room 1</td>
<td></td>
<td>Cindy Harris</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Judge A, B</td>
<td>Cindy Harris</td>
<td>Fred Harris</td>
</tr>
<tr>
<td>Room 2</td>
<td></td>
<td>Anyone</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Judge C, D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Room 3</td>
<td></td>
<td>Select Proctor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Judge A, B</td>
<td>Select Proctor</td>
<td></td>
</tr>
<tr>
<td>Room 4</td>
<td></td>
<td>Select Door Monitor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Judge C, D</td>
<td>Select Door Monitor</td>
<td></td>
</tr>
<tr>
<td>Room 5</td>
<td></td>
<td>Judge A, Judge B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Room 6</td>
<td></td>
<td>Judge A, Judge B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Illustration 16: Master Schedule document (.csv), listing all scheduler information.
<table>
<thead>
<tr>
<th>Yet Another Test</th>
<th>Format</th>
<th>Student Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saturday, Traditional</td>
<td>Niah</td>
<td>1</td>
</tr>
<tr>
<td>Traditional</td>
<td>Malac</td>
<td>1</td>
</tr>
<tr>
<td>Traditional</td>
<td>Kayla</td>
<td>1</td>
</tr>
<tr>
<td>Traditional</td>
<td>Sydni</td>
<td>1</td>
</tr>
<tr>
<td>Traditional</td>
<td>Christ</td>
<td>1</td>
</tr>
<tr>
<td>Traditional</td>
<td>Olivia</td>
<td>1</td>
</tr>
<tr>
<td>Traditional</td>
<td>Merel</td>
<td>1</td>
</tr>
<tr>
<td>Saturday, Traditional</td>
<td>Esper</td>
<td>2</td>
</tr>
<tr>
<td>Traditional</td>
<td>Haley</td>
<td>2</td>
</tr>
<tr>
<td>Traditional</td>
<td>Kyla</td>
<td>2</td>
</tr>
<tr>
<td>Traditional</td>
<td>Meag</td>
<td>2</td>
</tr>
<tr>
<td>Traditional</td>
<td>Lillian</td>
<td>2</td>
</tr>
<tr>
<td>Saturday, Traditional</td>
<td>May T</td>
<td>3</td>
</tr>
<tr>
<td>Traditional</td>
<td>Megh</td>
<td>3</td>
</tr>
<tr>
<td>Traditional</td>
<td>Chris</td>
<td>3</td>
</tr>
<tr>
<td>Traditional</td>
<td>Kai W</td>
<td>3</td>
</tr>
<tr>
<td>Traditional</td>
<td>Glynn</td>
<td>3</td>
</tr>
<tr>
<td>Saturday, Traditional</td>
<td>Gordo</td>
<td>4</td>
</tr>
<tr>
<td>Traditional</td>
<td>Claud</td>
<td>4</td>
</tr>
<tr>
<td>Traditional</td>
<td>Ana S</td>
<td>4</td>
</tr>
<tr>
<td>Traditional</td>
<td>Ashly</td>
<td>4</td>
</tr>
<tr>
<td>Sunday, 1 Traditional</td>
<td>Brade</td>
<td>1</td>
</tr>
<tr>
<td>Traditional</td>
<td>Mado</td>
<td>1</td>
</tr>
<tr>
<td>Traditional</td>
<td>Vishw</td>
<td>1</td>
</tr>
<tr>
<td>Sunday, 1 Non-Competitive</td>
<td>Miche</td>
<td>1</td>
</tr>
</tbody>
</table>

**Illustration 17:** Simplified Schedule document (.csv), listing basic student information.
**Illustration 18**: Teacher list document (.csv), listing basic teacher information.

<table>
<thead>
<tr>
<th>ID</th>
<th>Teacher</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Amy Jahr</td>
</tr>
<tr>
<td>2</td>
<td>Mayu No</td>
</tr>
<tr>
<td>5</td>
<td>Nancy Ku</td>
</tr>
<tr>
<td>12</td>
<td>Olga Barr</td>
</tr>
<tr>
<td>11</td>
<td>Cindy Ha</td>
</tr>
<tr>
<td>4</td>
<td>Kathleen</td>
</tr>
<tr>
<td>6</td>
<td>Kathie Kr</td>
</tr>
<tr>
<td>13</td>
<td>Christine</td>
</tr>
<tr>
<td>3</td>
<td>Rong Luc</td>
</tr>
<tr>
<td>7</td>
<td>Youseon</td>
</tr>
<tr>
<td>1</td>
<td>Savannah</td>
</tr>
<tr>
<td>8</td>
<td>Sonnet J</td>
</tr>
<tr>
<td>9</td>
<td>Farida Jai</td>
</tr>
</tbody>
</table>

**Illustration 19**: Theory scores document (.csv) column headers. Students not shown.

<table>
<thead>
<tr>
<th>Yet Another Test</th>
<th>Theory Score</th>
<th>Alternate Exam</th>
</tr>
</thead>
</table>
4 Testing

4.1 Acceptance Criteria

In order for ARIA to be a success, it must meet the following criterias which were set at the beginning of the development.

4.1.1 Registration

1. As an event administrator, I need a portal that will display event, teacher, and student information.
   - The portal is only accessible by a logged in administrator.
   - If the correct user credentials are not met, then the portal redirects to a login screen.
   - The portal displays a table containing sections for Events followed by the Teachers, Students, and Schedule information for said event.
   - The Events section shows the name, date, time and location of all entered events.
   - The Teachers section shows the names, contact information, and portal links of an event’s teachers.
   - The Students section shows the names, parent contact information, and registration status of an event’s students.
   - If no relevant data is available for any of the above-mentioned categories, then the table displays a notice to the user of this fact.

2. As an event teacher, I need a portal that will display a list of my students who have registered for the event.
● A teacher’s respective portal can only be accessed with the correct hash code.

● If the hash code is invalid, then an error page is displayed.

● The teacher portal displays a table containing sections for Events, Students, Schedules, and Tasks.

● The Events section shows the name, date, time and location of the teacher’s event.

● The Students section shows the names, parent contact information, and registration status of the teacher’s students.

● The Schedules section shows the scheduled times, location, and music pieces for the teacher’s students.

● The Tasks section shows the date, time, and location for a teacher’s assigned volunteer duties.

● If no relevant data is available for any of the above-mentioned categories, then the table displays a notice to the user of this fact.

4.1.2 Scheduling

1. As an event administrator, I need a schedule form so that I can create a schedule containing ample slots for scheduling students.

   ● The user completes all mandatory fields before submitting the form.

   ● The form contains appropriate values (e.g. numbers, words) in the appropriate input boxes.
● An error occurs when the provided information is insufficient to accommodate all students and student preferences.

● Upon a successful submission, the form must be stored in a scheduling database.

2. As an event administrator, I need a workable schedule interface because I may want to edit information on location, performance time, and student assignment.

    ● The user can change the order of student performance in a section.

    ● The user can move a student to perform in a different section.

    ● A warning message appears if student arrangement does not meet scheduling parameters or student preferences.

    ● Judges and volunteers can be assigned to each section.

    ● Section information (e.g. location, time) can be edited.

    ● The user can save the changes.

4.1.3 Document Generation

1. As an event administrator, I need an easy way to download various documents using the information gathered from the scheduler.

    ● The user can generate documents by clicking on a button found on the Administrator Portal.

    ● The user can download all documents, compressed in a ZIP folder format.

2. As an event administrator, I need a way to generate a wide variety of supporting documents that can be edited and printed at any time.
● The user can generate Announcing sheets, Adjudication forms, Result sheets, Teacher master page, Session assignments, and Room Schedule documents in an RTF format for Traditional and Master Class sections.

● The user can generate the Master schedule, Simplified schedule, Teacher list, and Theory score documents in a CSV format for Traditional and Master Class sections.

● The user can generate the Announcing sheet specific to the Command Performance.
4.2 Testing Workflow

All testing workflow can be validated with detailed use cases. They will be validated after use cases defined and tested.

4.2.1 Registration

4.2.1.1 Happy Path Workflow - Admin Portal

1. Administrator logs in and views Admin Portal.

2. The event’s Students section is displayed with the number of currently registered students and the table is populated with student information.

3. Navigate to the event’s Teacher section.

4. The number of registered teachers is displayed and the table is populated with teacher information.

5. Navigate to the Schedule section.

6. If an event has completed scheduling, then the table is populated with student schedule information.

7. Navigate to the Command Performance section.

8. If an event’s Command Performance has completed scheduling, then the table is populated with student Command Performance information.

9. Navigate to the Events section.

10. The number of events is displayed and the table is populated with all event information.

4.2.1.2 Happy Path Workflow - Teacher Registration

1. Teacher navigates to his/her teacher portal.
2. Navigate to Students section and click on “Continue with Registration” for a student.

3. A new tab is opened in the browser displaying the Teacher Registration page.

4. Fill out fields for required Volunteer Options.

5. Fill out fields for required Volunteer Times.

6. Make selections for student’s music performance pieces.

7. Fill out fields for student’s play duration and theory score.

8. Click the “Submit” button to complete Teacher Registration.

4.2.1.3 Negative Testing Workflow - Teacher Reg. with Invalid Fields

1. Navigate to a student’s Teacher Registration page.

2. Do not fill out Volunteer Options and click the “Submit” button.

3. Form displays an error, indicates to user of missing Volunteer Options selection, and retains all previously entered information.

4. Do not fill out Volunteer Times and click the “Submit” button.

5. Form displays an error, indicates to user of missing Volunteer Times selection, and retains all previously entered information.

6. Do not fill out Student music pieces composer and title fields, then click the “Submit” button.

7. Form displays an error, indicates to user of missing music piece selections, and retains all previously entered information (including any completed music piece selections).

8. Fill out theory score with a non-numeric value, then click the “Submit” button.
9. Form display an error, indicates to user of invalid field input, and retains all previously entered information.

4.2.2 Scheduling

4.2.2.1 Happy Path Workflow - Create a Schedule

1. Direct user to schedule creating module
2. Select name of event to schedule
3. Generate form to set scheduling parameters
4. Fill out scheduling form
5. Submit scheduling form
6. Store schedule form in the database
7. Generate schedule

4.2.2.2 Happy Path Workflow - View/Modify a Schedule

1. Direct user to schedule viewing module
2. Select name of event that has already been scheduled
3. Display current schedule onto the page
4. Modify student order/assignment
5. Edit section information
6. Save changes to the schedule

4.2.2.3 Negative Testing Workflow - Create a Schedule

1. Direct user to schedule creating module
2. Select name of event to schedule
3. Fill out scheduling form
4. Submit scheduling form

5. Generate error - the necessary information we not filled

6. Prompts user to fill out required information before resubmitting

7. Generate error - fields filled with inappropriate values

8. Prompts user to resubmit form with the appropriate values in the right field

9. Generate error - parameters given from the form cannot accommodate all registered students

10. Prompts user to adjust form inputs before resubmitting

4.2.3 Document Generation

4.2.3.1 Happy Path Workflow - Generate Documents

1. Administrator views Admin Portal.

2. The scheduler has been run, a schedule was generated successfully.

3. Administrator clicks the “Generate Documents” button.

4. A prompt appears, presses “OK” to download.

5. All files are downloaded by the browser, packed into a ZIP folder.

6. Administrator unpacks the folder, and views/edits the documents.

4.2.3.2 Negative Testing Workflow - Generate Doc. without Schedule

1. Administrator views Admin Portal.

2. The scheduler was not run, a schedule was not yet generated.

3. Administrator clicks the “Generate Documents” button.

4. An error message appears on screen.
5. User is informed that documents cannot be generated if a schedule has not been created.
5 Future Work

In terms of core functionality, ARIA has been fully implemented and can be used to support NNMTA in running the Reno Youth Music Festival from beginning to end. With the base completed, additional adjustments on ARIA can be made to meet the administrator’s requests and make the plugin more user friendly.

ARIA is currently designed to be used solely for the NNMTA music festivals. One direction for the application would be to extend its usability beyond its current scope. To achieve a more general music event manager, it is recommended to develop the project independent from WordPress and Gravity Forms. This would allow more flexibility in programming the software and allow more accessibility in the market.
Conclusion

ARIA is an event manager that assists NNMTA in running the Reno Youth Music Festival. It handles and processes the paperwork done in registration, scheduling, and document generation. As a result, running the music festival has become more efficient and less tedious and error prone. It has been fully implemented as a WordPress plugin, but developing the system independently from the platform would increase its flexibility and accessibility. In its current state, ARIA has become a reliable application with many potential areas to expand and improve.
Bibliography

   http://www.nnmta.org/events-2/reno-youth-music-festival/

   https://wordpress.org/about/

3. Gravity Forms, Form Builder Features (website).
   http://www.gravityforms.com/features/form-builder/

4. ARIA CSE Testing Site (website).
   https://aria.cse.unr.edu/

5. ARIA SiteGround Testing Site (website).
   http://aria-development.org/

6. RegPack (website)
   http://www.regpacks.com

7. RegFox (website)
   http://www.regfox.com

8. RegOnline (website)
   http://www.regonline.com
## Appendix

### A - Glossary

<table>
<thead>
<tr>
<th>#</th>
<th>Terms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ARIA</td>
<td>Administration, Registration, and Information Assistant</td>
</tr>
<tr>
<td>2</td>
<td>Gravity Forms</td>
<td>A Wordpress plugin that allows the creation of advanced forms with various data type fields.</td>
</tr>
<tr>
<td>3</td>
<td>WordPress</td>
<td>A CMS that operates on PHP and MySQL. It is free and open source.</td>
</tr>
<tr>
<td>4</td>
<td>CMS</td>
<td>Content Management System</td>
</tr>
<tr>
<td>5</td>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>6</td>
<td>PHP</td>
<td>PHP: Hypertext Preprocessor. It is a server-side scripting language.</td>
</tr>
<tr>
<td>7</td>
<td>JavaScript</td>
<td>An interpreted programming language that can be used client-side to modify a web page’s appearance or behavior.</td>
</tr>
<tr>
<td>8</td>
<td>HTML</td>
<td>HyperText Markup Language. The standard language for creating web pages and applications.</td>
</tr>
<tr>
<td>9</td>
<td>CSS</td>
<td>Cascading Style Sheets. Used to describe the presentation of a document written in a markup language.</td>
</tr>
<tr>
<td>10</td>
<td>PayPal</td>
<td>A company providing a worldwide online payments service.</td>
</tr>
<tr>
<td>11</td>
<td>PayPal IPN</td>
<td>PayPal Instant Payment Notification. This is a messaging service that alerts merchants of PayPal transactions.</td>
</tr>
<tr>
<td></td>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>---</td>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>12</td>
<td>WordPress Plugin</td>
<td>Packaged code that can be activated within Wordpress to add features and functions.</td>
</tr>
<tr>
<td>13</td>
<td>WordPress Form</td>
<td>A form whose fields and entries are linked to a Wordpress database.</td>
</tr>
<tr>
<td>14</td>
<td>Command Performance</td>
<td>The main performance for which distinguished students will be invited to attend.</td>
</tr>
<tr>
<td>15</td>
<td>Master Class</td>
<td>Extra-long sessions in the screening process, including a music lesson.</td>
</tr>
</tbody>
</table>