Abstract
We have provided an optimal scheduling system for the Suzuki Musical Institute at Dallas (SMID) to provide a better what parents can schedule their children based on the time slot available given by the teacher. The SMID Scheduling System project will include a web application for use of the Suzuki Music Institute of Dallas to automate scheduling of student lessons.

A successful project will minimize the amount of time an instructor will spend at the system until the final schedule is created, without limiting the freedom of instructors to modify the tentative schedule as they wish before it is finalized.

This web application will have a foundation of a backend system connected to a database for the saving of student information over time for the ultimate congregation of the data through a custom algorithm resulting in potential final schedules.

Results
The software provides an MVC architecture of the system. The system consists of 4 levels of layers, UI (front-end), back-end, DB and algorithm,

UI (Front-end):
- Javascript
- Admin View
- parent information
- student information
- adding/updating/viewing classes, roles, users & user roles
- Parent View
- add/delete a preferred schedule interval
- add comment to be sent to teacher

Back-end:
- Endpoints
- Users, Parent, Class, Role, Student, User Role, Email
- User Authentication/Authorization/Session/Role Based Authentication

DB:
- mysql - hosting many tables to hold the essential data
- tables including: class, final_schedule, preferred_schedule, role, student, user, userrole

Algorithm:
- OPL running CPLEX; the following constraints were used to generate a usable and efficient schedule:
  - ensure that no two students have lessons at the same time
  - ensure that the teacher is able to teach for the lesson scheduled
  - ensure that the student is also available for the scheduled lesson
  - ensure the students were scheduled for the full length of their lesson
  - ensure that parents are not required to travel to SMID more than a set tolerance dependent on their family size
- OPL uses SQL commands to import and export data to and from the database

The algorithm, with realistic test data, will generate a valid schedule in an estimated 2-5 seconds. The solution will best accommodate all parents schedule with the minimal time being speant at SMID as possible.

Architecture

GUI - Parent

Summary
Increasing efficiency for instructors of lesson scheduling was the primary goal. Allowing our algorithm to complete schedule creation, and user-interface to document parent input, is our way of automating this process. Our application has the foundational structure to allow for this result, with the hope of being effective at its original purpose after having a more polished user-interface, increased control over parents and instructors to input restrictions on the schedule, and increased functionality to allow for on-the-fly changes from instructors.

Two main aspects of our application characterize the success of the project, namely the CPLEX algorithm and the backend structure to save, format, and display data. While these aspects are present, the effectiveness of the application after deployment will be improved through user-interface improvements.

Ultimately, this project acted to provide valuable experience to the team in learning how to develop web-based software. This experience includes maintaining an effective development timeline, learning the languages used more in-depth, and thinking creatively in order to solve software problems. Our group would like to thank our advisors and professors for allowing us the freedom to learn more, even while making mistakes along the way.

Impact
Our project’s impact is to significantly reduce the amount of time that faculty members in educational environments (specifically the Suzuki Music Institute of Dallas) spend on the construction of schedules that accommodate individual student’s available time.

The linear programming approach considers all possible schedules, and generates the one that best fits with all the families enrolled. This approach utilizes the disregard for runtime efficiency since the schedule generation process needs to be only done once, and the users can afford to allow it to be run in the background.

The algorithm is generic enough to be adapted to similarly structured educational environments, or to be modified to improve how courses are scheduled in even a university setting, to accommodate when classes are available depending on demand by major or by other factors.