UCP Bailes Teacher Dashboard

Group 2

April 28th, 2016

Members
Mario Carrasco
Christopher LaChance
Lyudmila Sandomirskaya
Raphael Saint-Louis

Sponsor
UCP Bailes Elementary School
# Table of Contents

Executive Summary ........................................................................................................... 1  
Technical Content ........................................................................................................... 3  
Significance .................................................................................................................... 3  
Motivation ....................................................................................................................... 4  
    Mario Carrasco ............................................................................................................ 4  
    Christopher LaChance .............................................................................................. 4  
    Lyudmila Sandomirskaya ........................................................................................... 5  
    Raphael Saint-Louis .................................................................................................. 5  
Broader Impacts .............................................................................................................. 6  
Project Overview ........................................................................................................... 7  
    About UCP Bailes ..................................................................................................... 7  
    Functionality ............................................................................................................ 7  
Challenges ..................................................................................................................... 8  
Technical Objectives ..................................................................................................... 9  
    High-level Goals ..................................................................................................... 9  
    Requirements .......................................................................................................... 16  
Research ....................................................................................................................... 22  
    PHP: ....................................................................................................................... 22  
    Laravel Framework ................................................................................................. 24  
    MVC: ....................................................................................................................... 26  
    AngularJS ................................................................................................................ 27  
    MySQL .................................................................................................................... 30  
    MongoDB ................................................................................................................ 30  
    APIs .......................................................................................................................... 32  
    OAuth2.0 API ......................................................................................................... 36
# Table of Contents

- The Proposed System: Needs ................................................................. 91
- The Proposed System: Account Types and Permissions ....................... 92
- The Proposed System: Implementation ............................................... 94
- Project Plan ...................................................................................... 95
  - Process ....................................................................................... 95
  - Building ................................................................................... 95
- Testing .......................................................................................... 98
- Integration Testing ........................................................................ 100
- System Testing ............................................................................. 101
- Evaluation .................................................................................... 102
- Related Work .............................................................................. 103
  - Popular Learning Management Systems ........................................ 103
  - Common LMS Features ............................................................. 104
  - Currently Used Solutions .......................................................... 105
  - Issues with Current Solutions ..................................................... 108
- Facilities and Equipment ................................................................. 112
- Sponsor Interaction ....................................................................... 113
- Suppliers and Consultants .............................................................. 116
- Administrative Content ................................................................. 117
- Budget ......................................................................................... 117
- Milestones .................................................................................... 117
- Project Summary and Conclusions ............................................... 119
- References ................................................................................. 120
Executive Summary

With the advent of technology into classrooms across the world, and increasingly standard internet connectivity sweeping across homes and economic classes, educational institutions have adopted online utilities to complement the classroom curriculum. The massive amount of resources online is both a blessing and a burden; teachers have the freedom to pick and choose appropriate educational content for their classrooms and adapt to individual student needs, but often at the expense of having grades and performance statistics fragmented across many websites.

UCP East Orange Bailes Campus, also know as UCP Bailes Elementary, has been caught up in this exact predicament. Recent funding and e-learning initiatives have increased after-hours strain on the teachers and administrators, who now have to log onto a variety of websites, pull data from cloud services, and collaborate with other teachers to produce a simple progress report for each student. Because different classes may use different websites, the process is neither centralized nor standardized and can easily consume four-plus hours after classes have been dismissed for the day.

The main problem is that there is no central system used to gather information from all the supplementary websites and online activities, where the teacher can view a summary of the students progress.

Inherent problems are also present, as the school and curriculum is designed to support and enhance learning for children with disabilities of all types. Traditional letter grades and percentages do not accurately convey a student’s progress, as student performance can not be relative to other students. Instead, overall performance must be relative to their own progress and achievements over the course of the academic year. Therefore, hard grades (i.e. ‘A’, ‘B’, ‘C’, ‘D’, and ‘F’) based on percentages are meaningless and have virtually no indication of the child’s accomplishments and success in the class.
The purpose of this project is to approach this problem and develop a solution to remedy the current issues, streamlining the workflow of the teachers, so that ultimately the time spent doing dull, repetitive tasks can be automated and reclaimed as time for drafting activities and lesson plans for the children. In a broad sense, the goal is to create an online dashboard to retrieve the information from the websites used by each classroom, store the information in a database, and present a summary to the teacher and administrators.

This project would also focus highly on answering the question “How are the students performing?”. This is a question that is asked repeatedly by the teachers themselves, the administration at the school, the school board representatives, and most importantly, the concerned parents of these students. For as often as this question is asked, it has proven extremely difficult to answer. When a parent wants to know the progress of their child, the teacher must traverse a variety of files across multiple platforms, including the teachers personal hard drive, various google docs, the many websites used by the teacher to host tests and assignments, and many more. This is a very inefficient process that must be done on a very regular basis.

Our application will use various techniques to gather all of this student data, decide what is important, store it in a single place, and display it in a meaningful manner. The overall goal of this project is to allow the user to get a quick overview of the important student metrics to easily see how that student is doing. Then, allow the user to expand from the more broad metrics to get a more detailed report of the student performance. With all of this data so easily accessible (for users with the given permission), the application will essentially bridge the gap between the teachers and all parties interested in student performance, such as parents, school administration, and the school board. No longer will the teacher have to waste time addressing each client on a student by student basis. Instead, the teacher can easily post all student information for the permissible users to access.
Technical Content

Significance

This application is ultimately designed to assist with the intense day-to-day workload of the teachers and school administrators at the UCP Bailes Elementary school. Much of the time spent by teachers and administrators is managing massive amounts of data on a wide variety of platforms, and the many platforms only exponentially increases the challenge of data retrieval. This application will ease the process of managing all of this data, so that more time and energy can be spent on what really matters, the students.

UCP, or United Cerebral Palsy of Central Florida, is a series of charter schools created to provide hope for children with disabilities. Prior to UCP, very few options existed for these children [1]. Since its existence, UCP has begun accepting children with a wide range of disabilities, such as cerebral palsy, Down syndrome, autism, spina bifida, speech and hearing delays [1]. They have also opened their education program for children without disabilities. This creates an open environment for children to learn, play and socialize without limitations or fear of being excluded.

UCP’s mission statement is clear, “To empower children with and without disabilities to achieve their potential by providing individualized support, education and therapy services in an inclusive environment [2].” They add on to that by saying, “at UCP of Central Florida, we believe every child deserves academics to help them learn, grow and excel [2].”

This project is significant because it has been designed to support these goals. The project is explicitly intended to ease the more tedious tasks for the teachers and administrators, such as taking attendance and managing behavior-related forms. The project has also been created to solve a more difficult problem, managing various student and teacher data and congregating it in a single location. Combining these two critical features of the application, the teacher should be able to focus much more on the
students, and less on the data. The goal of this project is to allow UCP to better fulfill their mission statement by creating a more cohesive work environment.

**Motivation**

Mario Carrasco

I want to help others. I believe that everyone has a talent or ability that they can use to contribute to society and the well-being of their surrounding community, be it creating artwork or solving problems. I also believe that humans make the greatest advances by assisting each other, and that we are naturally inclined to work together to overcome barriers that cannot otherwise be avoided.

In that same context, my greatest mentors have been educators throughout my academic career. They have opened countless doors, inspired motivation, and shared guidance with me and dozens to hundreds of other students—all with relatively little compensation in return. The best instructors tend to expend gross amounts of time, resources, and energy outside official work hours, and do so selflessly.

Teachers are largely responsible for the mindset and attitude of every new generation, and so I hope to reimburse them in the form of this Senior Design collaboration with UCP Bailes Elementary. My hopes are that the day-to-day routine of the staff at UCP Bailes becomes incredibly simple, and that aggregating student data becomes trivial. This way, teachers can spend more time being tending to the needs of their students, expanding minds, and providing the attention and care that every child deserves.

Christopher LaChance

I am very excited to be working on this project for UCP Bailes for a number of reasons. The main reason that I chose this project as one of my top picks was because I strongly believe in the mission of UCP. I want children of all backgrounds, strengths, and weaknesses to have the ability to learn new things and accomplish great tasks. I understand that the
teachers at the school must put forth a lot of extra effort to give these students the attention that they need, but are struggling to find the time to both dedicate to the students and complete all of their required tasks. I would like to work very hard to ease the required tasks of the teachers, such as inputting and retrieving data about their students assignments and tests. The other reason why I chose this project is because there were a lot of different component involved in this project. For example, there must be a database and server that our web application interacts with. This will be a great experience for me to learn these technologies because I have not worked with these components on this scale before.

Lyudmila Sandomirskaya
The reason why UCP Bailes proposed project was my first choice is because I understood the difficulties that teachers and administrative personnel are facing in a day-to-day basis with multiple time consuming reporting tasks they need to accomplish, so I wanted to contribute my skills and time to build a system that would ease the reporting process leaving more time for teachers to spend on lesson preparation and other educational activities that will directly benefit their students and enrich educational process.

Raphael Saint-Louis
My initial attraction towards this project was due to it's practicality and it's possible usefulness for the clients requesting it. The project wasn't about entertainment, or information gathering, but about helping a local school with a very real and tangible problem. Because of this contrast in goals, I felt that this was the project in which my efforts would be most impactful, and that also made the project seem that much more interesting to me. Additionally, the timing of this project couldn't be more perfect, as my familiarity with servers and databases was something I was hoping to improve, and this project presented an amazing opportunity for me to learn and improve upon these skills. My hope is that our final project will be outstanding enough for other schools to adopt it, as helping as many instructors as possible with this project is me and my team's ultimate goal.
Broader Impacts

The project, at no cost to the school, aims to serve educators and administrators by allowing automation of trivial, but time-intensive, organizational tasks. UCP East Orlando / Bailes Campus is rapidly growing, and with an influx of students and demand for larger accommodations, teachers are increasingly strained with the task of maintaining records for each student, across the advanced technological platforms provided by the school, such as iPads and web applications.

The web application that is being designed will not only allow this particular school to continue using online technologies and methods to engage their students, but it aims to also provide a foundation for other schools and organizations to build upon. Whether public or private, if other institutions would like to adopt more modern methods of teaching, the project will be available free of charge to assist in allowing a school to perform at the highest level.

In an increasingly interconnected world, it’s important, no matter what a child’s future profession is, to be acquainted with technology and knowledgeable in how to navigate it with ease. This project, if successful, will open a gateway of opportunities for teachers to spend less time on the monotonous task of data administration and more time for creating engaging activities for students and collaboration with other educators and parents in their off-hours.

Education generally occupies the first eighteen years of an individual’s life. The earliest years tend to be the most sensitive to instilling a sense of motivation, encouragement, and achievement in a young generation. Educators literally shape our society, and have the power to change lives. Teachers provide mentorship to children who may not have the supportive guidance that every child deserves. It’s for this overlying reason that the team feels it holds the responsibility to give back to educational institutions whenever possible.
By providing our software for free, and open to change, our team will release a flexible, low-cost, scalable system to enhance the experience of educators and administrators seeking to ease an extraneous load that’s better left for automation.

**Project Overview**

**About UCP Bailes**

Officially known as UCP Bailes / East Campus, this non-for-profit charter-school offers education and services for students from kindergarten up to fifth grade. The primary focus of UCP Bailes is to service and integrate students with and without disabilities in order to maximize learning, collaboration, and necessary skills. In the context of the project at hand, the school:

- Caters to students “with a spectrum of disabilities”
- Is not included in county / state initiatives
- Needs a central ‘hub’ to view summary of student progress
- Wants an automated process for gathering student data

**Functionality**

The essential purpose of the system, as identified by UCP Bailes, includes:

- Manual entry of paper assignments
- Sharing of resources amongst staff
- Overview of student performance
- Processes to fill out a series of student behavior forms
Challenges

- Many websites, no API
  - Web-scraper / Importing data is difficult
  - Change in websites can break web-scraper
  - Future websites require manual entry

- Extracting relevant data
  - Filter out irrelevant data
  - Allow teacher to identify relevant data

- Automation may require authorization
  - Need to login to scrape

- Varying degrees of disabilities (none too severe)
  - Every student is different
  - Define "progress" -- not a level playing field
  - No known grading system accommodates current approach

- Lack of standardization
  - Everything copied by hand
  - Different websites for different teachers
  - Data / Record storage is mostly local

- Managing sensitive information
  - Student data must be securely stored
Technical Objectives

Goals

The basis of this project is to design a platform to serve as a content management system to easily store and access data for the staff at UCP Bailes Elementary School.

- This platform will include a centralized database to store information about the students at the school
- The platform will have a way to retrieve additional files and forms that the staff will need
- The platform must be easy to use and display information in a consistent and standardized format.

Our main focus for this project is to consolidate many forms of student evaluations and performance metrics from multiple sources including web sources, Excel spreadsheets, Google Docs, and hard copies of assignments. This student performance data shall be easily viewable for both teachers/school staff, and parents.

High-level Goals

- Simple, easy-to-use interface
- Export / Import student data
  - Import from websites and existing Excel sheets
  - Export database to CSV (comma-separated values) file
- Allow:
  - Access by administrators
  - Access by staff
  - Access by parents
- **Functionality:**
  - Manual entry of paper assignments
  - Sharing of resources amongst staff
  - Overview of student performance
  - Processes to fill out a series of student behavior forms
- **Inexpensive**
  - Cost = server + maintenance

**Objectives**

To manage all of the student performance data along with various teacher files, a web application will be created. This web application will contain a web portal, a web service, and a database. The web portal will be the interface for the various users. Here, users will be able to login to their accounts, interact with the interface, and retrieve the data they need.

- Create the web service
  - Database
- Storing all of the data for the school
  - Portal
    - providing the interface for the users
  - Web Server
    - Interacting with the database and updating the portal
- Retrieve student performance data from the external sources
  - Website scraping
  - Export file parsing
  - Manual entry
- Organize and display student information in an easy to find and easy to read format
- Manage levels of access for the various user types
- Provide the teachers with a common place to store and share their files
Objectives Elaboration

Objective: create a web service

One of our most basic objectives is to have a fully-functional web service or web portal. To complete that objective, we have to connect a secure database running on MySQL, a web portal displaying information via HTML, CSS, and Javascript, and a web server with a PHP backend interacting with our database.

Our database will, of course, contain all the student/parent as well as teacher information for the school. To store the initial information, we want to use websites with usable APIs to obtain and store the information in our database. The issue with this goal is that most of the websites do not have usable APIs, so we will have to rely on scraping the information from most of the websites by hand.

This can be done using a plethora of different web scrapers. So far, we’ve singled-out two possible web scrapers we may use, one of them depends on the chrome web browser (it’s a browser extension), and the other is a sophisticated python program made specifically for web scraping called “scrapy” (http://scrapy.org).

Once we gather the information that we’re able to, from all the websites that we’re able to, the final step is to actually input all the information in our database.

For the front-end of our program, we decided to keep it simple and keep it clean. There are two reasons for this:
1. If we have a nice, simple interface, it’ll be very easy to add and connect different logic to it at a later date.

2. The users of our program may not be very proficient with interfacing with computers, so for that reason, we want it to be so simple and easy-to-navigate so that even the least proficient computer users will be able to properly interact with our portal.

As a matter of fact, our team saw one of the websites that the teachers are currently using to grab information from, and our team thought it was fairly simple and straight-forward, but the teachers, our clients, did not feel the same way. They felt that it was too obtuse and convoluted, so we really do have to make this as simple as we can without excluding functionality, which will prove to be a challenge on its own.

Regarding the CSS of the program, we’ve decided to mostly rely on bootstrap for the stylesheets as none of our members have ever really used CSS before and we’d like to focus on the project at hand rather than spend valuable hours styling the web pages.

For our backend, we plan on using the Laravel framework (which is a popular MVC PHP framework). The reason we chose this is because a lot of the requirements of our app can be achieved more easily with some of laravel’s basic pre-configured settings. One example is that laravel comes bundled with security in mind, so the password hashes and salts come pre-secured for us. Another equally-important aspect of laravel is that it comes with pre-built bootstrap support, so we don’t have to really worry about setting that up.
**Objective:** Have different levels of access for the accounts

Separate levels of access when it comes to permissions is a very basic principle when it comes to security, both on the web and local machines. When it comes to our permission system, we modeled it a lot like the UNIX permission system. We decided on four levels of access: super-administrator, administrator, teacher, and parent. The reason we decided on having (seemingly) two top-level administrator accounts is because as we delved further and further into the inner-workings of our web service, we realized that it was necessary.

Our clients want there to be a way of representing the principal accounts. The reason for this is to have an account to represent the principal and vice-principal users, and those account types are represented with the administrator account. The administrator/principal account is able to view all the classrooms (and therefore all the students) in our database, but the administrators however, are NOT allowed to modify any of the information in the database.

The maintainers of the system (our group) are to be the only ones with full access to the system (can view every classroom, can add/remove classrooms and students, etc.). Once the program is deployed, the super-administrator accounts will be transferred over to the new maintainers of the server, which will be the IT guys responsible for the server.

The teacher account has the next highest permissions. The teacher account is able to change the grades and attendance of all the students that the teacher is in track of. So in this way, the permissions are not strictly linear, as the teacher account has write permissions to some things in the database (the teacher’s classrooms), while the administrator account has no write permissions anywhere, but has full read permissions for all classrooms and students.
The parent and student accounts are combined into one, so a parent with multiple students will have multiple accounts, one for each student. This account type only has read access, and only for the student in question. With that said, we haven’t actually finalized this part of the design as of yet, the alternative way is to have each parent account tied to as many student accounts as that parent has. This way may prove to be unnecessarily difficult, and if that’s the case, we’ll simply have a separate account for each student.

**Objective:** The teacher should be able to add performance metrics for their students

This objective relies on the students’ information being retrievable, which means that we need to have the web-scrapers working in order to achieve this objective. In addition to the web-scraping working effectively, we will also need to find a way to save the teachers’ session information on the different websites they visit, and then use that information to log-in behind the scenes so that we may have permission to gather student information from the websites. To begin with, there will be a button that can optionally be next to each student or next to each class (to update their information manually).

We will of course try and find a way to have the information gathered on a fixed schedule (maybe once per night) so that the teacher isn’t expected to always manually update their student information. Right now, we’re currently planning on relying on cookie information in order to retrieve the teacher’s credentials on other websites that they’re logged in, and if they logged out of a particular class, we plan on having a pop-up asking them to log-in. Hopefully this process isn’t too obtrusive as to detract too much from the user experience.
**Objective:** Teachers will be able to store private files to be accessed by other teachers

Since this web-app is meant to replace all the plethora of different websites that the teachers currently use, then we need to find a way for the teachers to have a drop-box like way to link other teachers to private files without needing to use email. Currently, the teachers use google docs in order to share links and different files with each other. The problem with this is that it creates a situation in which a lot of different links are being passed between the teachers with no way of really tying it back to whoever posted it or whether or not the link is even relevant anymore. We plan on having a comment section alongside the content area in which the links are shared, so teachers are able to comment to the links and/or files that are posted.

**Objective:** The system shall provide a way for teachers to input student information

The core requirement to mark this objective as complete is to finish the teacher dashboard. The teacher dashboard is the cumulation of all the different views and functions we'd like the teacher to have. First of all, teachers will be presented with an initial view of their dashboard. From there, they are able to navigate to their different classes. From a particular class view, the teachers are then able to see all the students they have in that class. Alternatively, teachers will also have a student view that displays information pertaining to all of their classes, perhaps with a column indicating that student’s class information.

Anyway, once a teacher clicks on a particular student, the student view will be invoked. In the student view for the teacher, one is able to see the grades of the student for each of the different classes that the student is involved in AND that the particular teacher has access to view. It is in this view that the teacher will be able to press a button to grab/update the student’s grade information from the different web sources that the website interfaces with.
## Security Requirements

### Users

<table>
<thead>
<tr>
<th>No.</th>
<th>Statement</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.01</td>
<td>The system shall have 4 levels of access:</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>• Super Administrator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Administrator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Teacher</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Parent</td>
<td></td>
</tr>
<tr>
<td>1.02</td>
<td>The highest level of access is Super Administrator (Super Admin)</td>
<td>No</td>
</tr>
<tr>
<td>1.03</td>
<td>The Super Administrator shall be able to view and edit all records in the database as well as perform any actions that the lower levels of access can perform</td>
<td>No</td>
</tr>
<tr>
<td>1.04</td>
<td>The second level of access is Administrator (Admin)</td>
<td>No</td>
</tr>
<tr>
<td>1.05</td>
<td>An administrator in the system shall be able to view all records in the database</td>
<td>No</td>
</tr>
<tr>
<td>1.06</td>
<td>The third level of access is Teacher</td>
<td>No</td>
</tr>
<tr>
<td>1.07</td>
<td>The teacher shall be able to update their current student’s grade</td>
<td>No</td>
</tr>
<tr>
<td>1.08</td>
<td>The teacher shall be able to create a new class and add students to it</td>
<td>No</td>
</tr>
<tr>
<td>1.09</td>
<td>The teacher shall be able to add parent accounts for their class</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Result</td>
</tr>
<tr>
<td>---</td>
<td>-------------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>1.10</td>
<td>The teacher shall be able to add performance metrics for their students</td>
<td>No</td>
</tr>
<tr>
<td>1.11</td>
<td>The teacher shall be able to store private files that can easily be shared among staff</td>
<td>Yes</td>
</tr>
<tr>
<td>1.12</td>
<td>The fourth level of access is the Parent</td>
<td>No</td>
</tr>
<tr>
<td>1.13</td>
<td>Each student at the school has a single corresponding parent account that is used to access the student’s performance metrics</td>
<td>No</td>
</tr>
</tbody>
</table>
### User Login

<table>
<thead>
<tr>
<th>No.</th>
<th>Statement</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.14</td>
<td>The system shall have a login with a username and password</td>
<td>No</td>
</tr>
<tr>
<td>1.15</td>
<td>The password must be at least 6 characters with at least one uppercase, one lowercase, and one special character</td>
<td>No</td>
</tr>
<tr>
<td>1.16</td>
<td>The passwords will be encrypted before they are stored in our database.</td>
<td>No</td>
</tr>
</tbody>
</table>

### Data Storage and Gathering Requirements

#### Student Data Storage and Access

<table>
<thead>
<tr>
<th>No.</th>
<th>Statement</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.01</td>
<td>Each student in the system will have a report of their grades for each subject they are in</td>
<td>No</td>
</tr>
<tr>
<td>2.02</td>
<td>Each student in the system will have performance metrics that have been gathered from the programs that their teachers use</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.03</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.04</td>
</tr>
</tbody>
</table>
## Gathering Student Information

<table>
<thead>
<tr>
<th>No.</th>
<th>Statement</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.05</td>
<td>The application shall list all of the supported programs that teachers can use to gather student performance metrics. The program is considered supported if the system can store, retrieve, and display the data for that program</td>
<td>No</td>
</tr>
<tr>
<td>2.06</td>
<td>For programs that generate some output file specifying student performance, the teacher shall be able to add that file to our system to be parsed</td>
<td>Yes</td>
</tr>
<tr>
<td>2.07</td>
<td>Our system shall be able to directly inspect and analyze some supported programs to retrieve data regarding student performance</td>
<td>Yes</td>
</tr>
<tr>
<td>2.08</td>
<td>The teacher shall be able to manually enter data gathered by supported programs to create a report of their students</td>
<td>No</td>
</tr>
</tbody>
</table>
## Teacher Data Storage

<table>
<thead>
<tr>
<th>No.</th>
<th>Statement</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.09</td>
<td>The teacher shall be able to store and share links</td>
<td>No</td>
</tr>
<tr>
<td>2.10</td>
<td>The teacher shall be able to store and share files of these types: .xls, .xlsx, .docx, .ppt, .pptx, and .txt</td>
<td>Yes</td>
</tr>
<tr>
<td>2.11</td>
<td>The teacher shall be able to choose what other teachers (if any) they would like to share a file with</td>
<td>No</td>
</tr>
<tr>
<td>2.12</td>
<td>The teacher will not be able to store any data that violates the HIPPA and COPPA laws</td>
<td>No</td>
</tr>
</tbody>
</table>
Research

PHP:

This server-side scripting language is the core of the entire backend for our project, but none of us had ever used it prior to starting with this project. PHP was initially created by Rasmus Lerdorf as a series interface programs in C to help him maintain his personal web page (hence PHP as an acronym initially standing for Personal Home Page), but as its popularity grew, it quickly evolved into a fully-fledged programming language. For our project, our group chose to work with PHP 5 rather than the newer PHP 7 (2004 release date versus 2015 release date) because of the greater support available for PHP 5 compared to 7.

What separates PHP apart from lot of other scripting languages, including JavaScript, is that the code is actually executed on the server, and from there the code is used to generate an HTML web page which is then sent to the server. That reason is why PHP is referred to as a server-side scripting language while languages such as (most) JavaScript are referred to as client-side scripting languages. The primary callers of client-side scripts are web browsers, which facilitate the processing of the script on the client’s computer. Remote scripts written in a client-side language must first be received from the web server and saved in a temporary location for the computer user where it’ll be run straight from the web browser.

Originally, our group planned on using pure PHP for the entirety of the backend of the project, but as our collective understanding of PHP improved, we decided it was a better idea to use the Laravel PHP framework as our backend language rather than raw PHP. The reason we chose Laravel rather than some other PHP framework is because it has a large community following, it promotes beautiful/organized code, it’s an MVC framework (more on this later) and it allows for template inheritance via its Blade Templating system.
One of the reasons we chose PHP is because our team needed some language that is easy to learn in short period of time, since we planned to start development by the end of the semester, and PHP appeared to be more straightforward to learn with our previous programming background, in comparison to javascript, for instance. Nevertheless, if time allows, our team will try to develop another version of the same web application using one of the javascript frameworks, most likely AngularJS. It would be a good learning experience where we can compare two different frameworks in action and encompass all the difficulties and benefits of using them.
Laravel Framework

Compared to other PHP frameworks, Laravel is quite new, released just in 2011. Despite this, Laravel is quickly becoming one of the most popular PHP frameworks. Laravel's popularity can be attributed to it's elegance; simply put, Laravel promotes extremely beautiful, simple, and easy-to-read PHP code. Another aspect contributing to its popularity is its modularity, thanks to composer, its package manager. Its relationship to composer allows the entire install to be configured, giving the user the choice to install only the certain parts of Laravel that they require. In addition to everything that's already been said, Laravel is also an MVC PHP framework, which seemed like a good idea based on the fact that different members of our group would be working on different parts at different times.

Figure 1. Components of Laravel
In Laravel you can store HTML (or a template that would be compiled into HTML behind the scenes) outside of the application’s logic, and just feed that template with the data that you retrieve from your application (“Templates in Laravel, Blade”).

The use of Blade Templates pretty much makes trivialized templates, and ensures that there will be minimal code duplication. This is because instead of creating several HTML pages that are nearly identical, except maybe for one line being different, for example, the only difference between a teacher’s dashboard and a principal’s dashboard may be that a principal has an extra content bar. With blades, we can have both pages use the same template, but then for one of the pages, we append other content.

Figure 2. Templates in Laravel, Blade.
MVC:

The MVC architectural framework separates a program into three primary components: the model, the view, and the controller. The model contains the program’s logic. In this case, the model will retrieve and store data into the database. The view is the outward representation of the application. The basic UI of the program will be represented by the view, so views will mostly be implemented with HTML and CSS. The controller handles user interaction, and will therefore connect the user’s button presses and actions on the view with the functions and database modification-capabilities of the model. The primary benefit of the MVC model is that it totally encapsulates the controller, the view, and the model from each other, so if a developer changes the layout of the view around, a different
developer working on the controller won't even necessarily notice that that's been changed, as it won't affect his work on the controller at all.

![Model = HTML

\textit{css Zen Garden}

\textit{The Beauty of CSS Design}

A demonstration of what can be accomplished this page.

Download the sample html file and css file

\textit{The Road to Enlightenment}

Littering a dark and dreary road lay the past n

Today, we must clear the mind of past practices:
W3C, Wasp, and the major browser creators

The css Zen Garden invites you to relax and re

![View = CSS

body {
  font: 12px/16px arial, helvetica
  color: #555;
  background: url(bg_left.gif) 
  margin: 0;
  padding: 0;
}

a {
  text-decoration: none;
  font-weight: bold;
  color: #655;
}

a:hover {
  text-decoration: none;
  font-weight: bold;
  color: #665;
}

![Controller = Browser

Figure 4. CSS styling.

A simple, yet not-often used example of the MVC pattern in action is a simple HTML page with some CSS stylings (in another file rather than being embedded in the HTML file). The controller of course, is the web browser. The styling of the page (CSS) can be said to represent the view, and the user will navigate on the page using their mouse or keyboard, and click on certain HTML elements on the page. This interaction compromises the most unsophisticated view-controller-model relationship that makes up the MVC model. As far as our project, instead of working just with plain html, css, and php, we will use a framework that implements MVC along with other features that will ease the development process.

\textbf{AngularJS}

AngularJS is an open-source web application framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model–view–controller (MVC) and model–view–viewmodel (MVVM) architectures,
along with components commonly used in rich Internet applications ("AngularJS").

AngularJS, along with javascript, are gaining more popularity due to such features as data binding, built-in extensive form validation, promises, testing, and many more features.

Figure 2. AngularJS - MVC framework.
Figure 3. Features of AngularJS.
MySQL

MySQL is an open-source relational database management system (RDBMS). In July 2013, it was the world's second most widely used RDBMS, and the most widely used open-source client–server model RDBMS. It is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open-source web application software stack (and other "AMP" stacks) ("MySQL"). The use of software is free under the GNU General Public License, there is a vast amount of documentation, tutorials, and discussion forums that cover and answer most of the questions and issues that can possibly raise with the usage of MySQL database.

MongoDB

MongoDB is a NoSQL database that stores records in Binary JSON format. It provides fast retrieval of records in comparison to RDBMS, but it does not handle joins securely. When database schema designed with NoSQL database in mind, then joins could be performed at the backend level, or simply avoided - depending on the type of application that uses the database.
In MongoDB tables replaced by collections, table rows by documents, and columns by fields (Figure 4, Figure 5).

**MongoDB Components**

- Actual data
- Needs RAM + Disk IO

- Stores sharding configuration
- Stores small amounts of data
- Infrequently

- Can run as Arbiter
- No data
- Just votes to elect primary

- Stateless router
- Typically run on App Servers

Figure 4. MongoDB components.
Figure 5. SQL vs NoSQL

APIs

Google Classroom API

We are looking into integrating Google Classroom into our web application. This will require OAuth 2.0 authentication to obtain permission to view classes from teachers. Because we will not have a level access as domain administrators, our access to classroom data will be limited and information such as personal student data will not be retrievable for our application. We will be able to view and modify classes and roosters and implement “Classroom share button” that will allow students and teachers share some content from our application back to Google classroom.
Below is a brief description of methods available through API calls.

### Table 1: Google classroom API - Collection `v1.courses`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| **create** | POST /v1/courses  
Creates a course. |
| **delete** | DELETE /v1/courses/{id}  
Deletes a course. |
| **get** | GET /v1/courses/{id}  
Returns a course. |
| **list** | GET /v1/courses  
Returns a list of courses that the requesting user is permitted to view, restricted to those that match the request. |
| **patch** | PATCH /v1/courses/{id}  
Updates one or more fields in a course. |
| **update** | PUT /v1/courses/{id}  
Updates a course. |

### Table 2: Google classroom API - Collection `v1.courses.aliases`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| **create** | POST /v1/courses/{courseId}/aliases  
Creates an alias for a course. |
| **delete** | DELETE /v1/courses/{courseId}/aliases/{alias}  
Deletes an alias of a course. |
| **list** | GET /v1/courses/{courseId}/aliases  
Returns a list of aliases for a course. |
### Table 3: Google classroom API - Collection `v1.courses.students`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| **create** | POST /v1/courses/{courseId}/students  
  Adds a user as a student of a course. |
| **delete** | DELETE /v1/courses/{courseId}/students/{userId}  
  Deletes a student of a course. |
| **get** | GET /v1/courses/{courseId}/students/{userId}  
  Returns a student of a course. |
| **list** | GET /v1/courses/{courseId}/students  
  Returns a list of students of this course that the requester is permitted to view. |

### Table 4: Google classroom API - Collection `v1.courses.teachers`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| **create** | POST /v1/courses/{courseId}/teachers  
  Creates a teacher of a course. |
| **delete** | DELETE /v1/courses/{courseId}/teachers/{userId}  
  Deletes a teacher of a course. |
| **get** | GET /v1/courses/{courseId}/teachers/{userId}  
  Returns a teacher of a course. |
| **list** | GET /v1/courses/{courseId}/teachers  
  Returns a list of teachers of this course that the requester is permitted to view. |
Table 5: Google classroom API - Collection v1.invitations

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| accept | POST /v1/invitations/{id}:accept  
Accepts an invitation, removing it and adding the invited user to the teachers or students (as appropriate) of the specified course. |
| create | POST /v1/invitations  
Creates an invitation. |
| delete | DELETE /v1/invitations/{id}  
Deletes an invitation. |
| get    | GET /v1/invitations/{id}  
Returns an invitation. |
| list   | GET /v1/invitations  
Returns a list of invitations that the requesting user is permitted to view, restricted to those that match the list request. |

Table 6: Google classroom API - Collection v1.userProfiles

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| get    | GET /v1/userProfiles/{userId}  
Returns a user profile. |
OAuth2.0 API

Access to Google Classroom will require the use of OAuth2.0 authentication protocol. Below is the documentation from www.developers.google.com/identity/protocols/OAuth2 that explains the steps to get authenticated.

1. Obtain OAuth 2.0 credentials from the Google Developers Console.

Visit the Google Developers Console to obtain OAuth 2.0 credentials such as a client ID and client secret that are known to both Google and your application. The set of values varies based on what type of application you are building. For example, a JavaScript application does not require a secret, but a web server application does.

2. Obtain an access token from the Google Authorization Server.

Before your application can access private data using a Google API, it must obtain an access token that grants access to that API. A single access token can grant varying degrees of access to multiple APIs. A variable parameter called scope controls the set of resources and operations that an access token permits. During the access-token request, your application sends one or more values in the scope parameter.

There are several ways to make this request, and they vary based on the type of application you are building. For example, a JavaScript application might request an access token using a browser redirect to Google, while an application installed on a device that has no browser uses web service requests.

Some requests require an authentication step where the user logs in with their Google account. After logging in, the user is asked whether they are willing to grant the permissions that your application is requesting. This process is called user consent.

If the user grants the permission, the Google Authorization Server sends your application an access token (or an authorization code that your application can use
to obtain an access token). If the user does not grant the permission, the server returns an error.

It is generally a best practice to request scopes incrementally, at the time access is required, rather than up front. For example, an app that wants to support purchases should not request Google Wallet access until the user presses the “buy” button; see Incremental authorization.

3. Send the access token to an API.

After an application obtains an access token, it sends the token to a Google API in an HTTP authorization header. It is possible to send tokens as URI query-string parameters, but we don't recommend it, because URI parameters can end up in log files that are not completely secure. Also, it is good REST practice to avoid creating unnecessary URI parameter names.

Access tokens are valid only for the set of operations and resources described in the scope of the token request. For example, if an access token is issued for the Google+ API, it does not grant access to the Google Contacts API. You can, however, send that access token to the Google+ API multiple times for similar operations.

4. Refresh the access token, if necessary.

Access tokens have limited lifetimes. If your application needs access to a Google API beyond the lifetime of a single access token, it can obtain a refresh token. A refresh token allows your application to obtain new access tokens.

Scenarios

Web server applications

The Google OAuth 2.0 endpoint supports web server applications that use languages and frameworks such as PHP, Java, Python, Ruby, and ASP.NET.

The authorization sequence begins when your application redirects a browser to a Google URL; the URL includes query parameters that indicate the type of access being requested. Google handles the user authentication, session selection, and user consent. The result is an authorization code, which the application can exchange for an access token and a refresh token.
The application should store the refresh token for future use and use the access token to access a Google API. Once the access token expires, the application uses the refresh token to obtain a new one.

For details, see *Using OAuth 2.0 for Web Server Applications*.

Token expiration

You should write your code to anticipate the possibility that a granted token might no longer work. A token might stop working for one of these reasons:

- The user has revoked access.
- The token has not been used for six months.
- The user changed passwords and the token contains Gmail, Calendar, Contacts, or Hangouts scopes.
- The user account has exceeded a certain number of token requests.

There is currently a limit of 25 refresh tokens per user account per client. If the limit is reached, creating a new token automatically invalidates the oldest token without warning. This limit does not apply to service accounts.
There is also a larger limit on the total number of tokens a user account or service account can have across all clients. Most normal users won't exceed this limit but a developer's test account might.

Using OAuth 2.0 for Web Server Applications

Creating web application credentials

All web applications that use OAuth 2.0 must have credentials that identify the application to the OAuth 2.0 server. Applications that have these credentials can access the APIs that you enabled for your project.

To obtain web application credentials for your project, complete these steps:

1. Open the Credentials page.
2. If you haven't done so already, create your OAuth 2.0 credentials by clicking Create new Client ID under the OAuth heading. Next, look for your application's client ID and client secret in the relevant table.
3. You can also create and edit redirect URIs from this page, by clicking a client ID. Redirect URIs are the URIs to your application's auth endpoints, which handle responses from the OAuth 2.0 server. You must change this value from the default example to the URI of your application's auth endpoint before you can use OAuth 2.0. For testing, you can specify URIs that refer to the local machine, such as http://localhost:8080. You should design your app's auth endpoints in a way that doesn't expose authorization codes to other resources on the page.

Download the client_secrets.json file and securely store it in a location that only your application can access.

Preparing to start the OAuth 2.0 flow

If you are using a Google API client library to handle the OAuth 2.0 flow, configure the client object, which you will use to make OAuth 2.0 requests. If you are handling the flow by directly accessing the OAuth 2.0 endpoints, just take
note of the client ID that you created in the previous step and the scopes you need to request.

To configure the client object:

Use the client_secrets.json file that you created to configure a client object in your application. When you configure a client object, you specify the scopes your application needs to access, along with the URL to your application's auth endpoint, which will handle the response from the OAuth 2.0 server.

For example, to request read-only access to a user's Google Drive:

```php
$client = new Google_Client();
$client->setAuthConfigFile('client_secrets.json');
$client->addScope(Google_Service_Drive::DRIVE_METADATA_READONLY);
$client->setRedirectUri('http://'.$_SERVER['HTTP_HOST'].'/oauth2callback.php');
```

Your application uses the client object to perform OAuth 2.0 operations, such as generating authorization request URLs and applying access tokens to HTTP requests.

**Redirecting to Google's OAuth 2.0 server**

When your application needs to access a user's data, redirect the user to Google's OAuth 2.0 server.

1. Generate a URL to request access from Google's OAuth 2.0 server:
2. $auth_url = $client->createAuthUrl();
3. Redirect the user to $auth_url:
4. header('Location: ' . filter_var($auth_url, FILTER_SANITIZE_URL));

Google's OAuth 2.0 server will authenticate the user and obtain consent from the user for your application to access the requested scopes. The response will be sent back to your application using the redirect URL you specified.
Handling the OAuth 2.0 server response

The OAuth 2.0 server responds to your application's access request by using the URL specified in the request.

If the user approves the access request, then the response contains an authorization code. If the user does not approve the request, the response contains an error message. All responses are returned to the web server on the query string, as shown below:

An error response:

https://oauth2-login-demo.appspot.com/auth?error=access_denied

An authorization code response:

https://oauth2-login-demo.appspot.com/auth?code=4/P7q7W91a-oMsCeLvLaQm6bTrgtp7

After the web server receives the authorization code, it can exchange the authorization code for an access token.

To exchange an authorization code for an access token, use the authenticate method:

```
$client->authenticate($_GET['code']);
```

You can retrieve the access token with the getAccessToken method:

```
$access_token = $client->getAccessToken();
```

Calling Google APIs

Use the access token to call Google APIs by completing the following steps:

1. If you need to apply an access token to a new Google_Client object—for example, if you stored the access token in a user session—use the setAccessToken method:
   ```
   $client->setAccessToken($access_token);
   ```
2. Build a service object for the API that you want to call. You build a a service object by providing an authorizedGoogle_Client object to the constructor for the API you want to call. For example, to call the Drive API:
   ```
   $drive_service = new Google_Service_Drive($client);
   ```
5. Make requests to the API service using the interface provided by the service object. For example, to list the files in the authenticated user's Google Drive:
   
   ```python
   $files_list = $drive_service->files->listFiles(array())->getItems();
   ```

   **Complete example**

   The following example prints a JSON-formatted list of files in a user's Google Drive after the user authenticates and gives consent for the application to access the user's Drive files.

   To run this example:

   1. In the Developers Console, add the URL of the local machine to the list of redirect URLs. For example, add `http://localhost:8080`.
   2. Create a new directory and change to it. For example:
      ```bash
      mkdir ~/php-oauth2-example
      cd ~/php-oauth2-example
      ```
   3. Clone the Google API Client Library for PHP:
      ```bash
      git clone -b v1-master https://github.com/google/google-api-php-client
      ```
   4. Create the files `index.php` and `oauth2callback.php` with the content below.
   5. Run the example with a web server configured to serve PHP. If you use PHP 5.4 or newer, you can use PHP's built-in test web server:
      ```bash
      php -S localhost:8080 ~/php-oauth2-example
      ```

   **Index.php**

   ```php
   <?php
   require_once 'google-api-php-client/autoload.php';
   session_start();
   $client = new Google_Client();
   $client->setAuthConfigFile('client_secrets.json');
   $client->addScope(Google_Service_Drive::DRIVE_METADATA_READONLY);
   if (isset($_SESSION['access_token']) &amp;&gt; $_SESSION['access_token']) {
   ```
$client->setAccessToken($_SESSION['access_token']);

$drive_service = new Google_Service_Drive($client);

$files_list = $drive_service->files->listFiles(array())->getItems();

echo json_encode($files_list);

} else {

$redirect_uri = 'http://' . $_SERVER['HTTP_HOST'] . '/oauth2callback.php';

header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));

}

Oauth2callback.php

<?php

require_once 'google-api-php-client/autoload.php';

session_start();

$client = new Google_Client();

$client->setAuthConfigFile('client_secrets.json');

$client->setRedirectUri('http://' . $_SERVER['HTTP_HOST'] . '/oauth2callback.php');

$client->addScope(Google_Service_Drive::DRIVE_METADATA_READONLY);

if (!isset($_GET['code'])) {

$auth_url = $client->createAuthUrl();

header('Location: ' . filter_var($auth_url, FILTER_SANITIZE_URL));

}
} else {

$client->authenticate($_GET['code']);

$_SESSION['access_token'] = $client->getAccessToken();

$redirect_uri = 'http://' . $_SERVER['HTTP_HOST'] . '/';

header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));

}

**Incremental authorization**

In the OAuth 2.0 protocol, your app requests authorization to access resources which are identified by scopes, and assuming the user is authenticated and approves, your app receives short-lived access tokens which let it access those resources, and (optionally) refresh tokens to allow long-term access.

It is considered a best user-experience practice to request authorization for resources at the time you need them. For example, an app that lets people sample music tracks and create mixes might need very few resources at sign-in time, perhaps nothing more than the name of the person signing in. However, saving a completed mix would require access to their Google Drive. Most people would find it natural if they only were asked for access to their Google Drive at the time the app actually needed it.

In this case, at sign-in time the app might request the profile scope to perform basic sign-in, and then later request the `https://www.googleapis.com/auth/drive.file` scope at the time of the first request to save a mix.

Using the procedures described in Using OpenID Connect and Using OAuth 2.0 to Access Google APIs would normally result in your app having to manage two different access tokens.

To avoid this complexity, you can include previously granted scopes in your authorization requests. For example:

$client->setIncludeGrantedScopes(true);
When you make an authorization request with granted scopes included, the Google authorization server rolls the authorization request together with all the previous authorizations granted to the requesting user from the requesting app.

Offline access

In some cases, your application might need to access a Google API when the user is not present. Examples of this include backup services and applications that make Blogger posts exactly at 8am on Monday morning. This style of access is called offline, and web server applications can request offline access from a user. The normal and default style of access is called online.

If your application needs offline access to a Google API, set the API client's access type to offline:

```php
$client->setAccessType("offline");
```

After a user grants offline access to the requested scopes, you can continue to use the API client to access Google APIs on the user's behalf when the user is offline. The client object will refresh the access token as needed.

Revoking a token

In some cases a user may wish to revoke access given to an application. A user can revoke access by visiting Account Settings. It is also possible for an application to programmatically revoke the access given to it. Programmatic revocation is important in instances where a user unsubscribes or removes an application. In other words, part of the removal process can include an API request to ensure the permissions granted to the application are removed.

To programmatically revoke a token, call `revokeToken()`:

```php
$client->revokeToken();
```
Detailed Design Content

- Design
  - Charts / Diagrams
  - Design choices / Tools
    - Server
    - Framework
    - Languages
    - Front-End
  - **Reason for choosing our development tools**
    - Overall Expectation of Usage
      - Load on server
      - Security concerns
      - Supported Browsers/Devices
  - **Advantages**
  - **Disadvantages**
Figure 3. High level design diagram.
Figure 4. Original ER diagram.
ER Diagram of created Database

Figure 5. ER diagram of created database.
Database creation script

Our implementation started with writing a database creation script that will create a database schema and would be run from MySQL Workbench or PhpMyAdmin (these are two programs that our team using for working with MySQL database)

# SQL commands to create and populate the MySQL database for senior
#design project COP4934 - Spring 2016

# delete the database if it already exists
drop database if exists ucpBailes;

# create a new database named ucpBailes
cREATE DATABASE ucpBailes;

# switch to the new database
use ucpBailes;

# create the schemas for the four relations in this database
create table user ( 
    userID integer AUTO_INCREMENT not null, 
    firstName varchar(20) not null, 
    lastName varchar(20) not null, 
    userName varchar(20) not null,
password varchar(20) not null,
primary key (userID)
) engine=InnoDB;

create table teacher (  
    teacherID integer AUTO_INCREMENT not null,
    userID integer not null,
    primary key (teacherID),
    foreign key (userID) references user(userID)
) engine=InnoDB;

create table student (  
    studentID integer AUTO_INCREMENT not null,
    userID integer not null,
    primary key (studentID),
    foreign key (userID) references user(userID)
) engine=InnoDB;

create table admin (  
    adminID integer AUTO_INCREMENT not null,
    primary key (adminID),
    foreign key (adminID) references user(userID)
) engine=InnoDB;

create table superAdmmin (  
    sAdmminID integer AUTO_INCREMENT not null,
primary key (sAdminID),
foreign key (sAdminID) references user(userID)
) engine=InnoDB;

create table teacherCredentials (  
    credentialID integer AUTO_INCREMENT not null,  
    teacherID integer not null,  
    userName varchar(20) not null,  
    password varchar(20) not null,  
    primary key (credentialID),  
    foreign key (teacherID) references teacher(teacherID)  
) engine=InnoDB;

create table class (  
    classID integer AUTO_INCREMENT not null,  
    className varchar(20) not null,  
    classDescription varchar(100),  
    primary key (classID)  
) engine=InnoDB;

create table subject (  
    subjectID integer AUTO_INCREMENT not null,  
    subjectName varchar(20) not null,  
    subjectDescription varchar(100),  
    teacherID integer not null,  
    foreign key (teacherID) references teacher(teacherID)  
) engine=InnoDB;
primary key (subjectID),
foreign key (teacherID) references teacher(teacherID)
) engine=InnoDB;

create table ThirdPartyApp (  
    programID integer AUTO_INCREMENT not null,  
    url varchar(200) not null,  
    appDescription varchar(100),  
    primary key (programID)  
) engine=InnoDB;

create table test (  
    testID integer AUTO_INCREMENT not null,  
    testName varchar(20) not null,  
    testDescription varchar(100),  
    aveClassScore integer,  
    programID integer,  
    subjectID integer not null,  
    primary key (testID),  
    foreign key (programID) references thirdPartyApp(programID),  
    foreign key (subjectID) references subject(subjectID)  
) engine=InnoDB;

create table testScore (  
    testID integer not null,  
    testName varchar(20) not null,  
    testDescription varchar(100),  
    aveClassScore integer,  
    programID integer,  
    subjectID integer not null,  
    primary key (testID),  
    foreign key (programID) references thirdPartyApp(programID),  
    foreign key (subjectID) references subject(subjectID)  
) engine=InnoDB;
create table testScore (  
testScoreID integer AUTO_INCREMENT not null,  
score integer,  
studentID integer not null,  
foreign key (studentID) references student(studentID),  
primary key (testScoreID)  
) engine=InnoDB;

create table assignment (  
assignmentID integer AUTO_INCREMENT not null,  
assignmentName varchar(20) not null,  
assignmentDescription varchar(100),  
averageClassScore integer,  
programID integer,  
primary key (assignmentID),  
foreign key (programID) references thirdPartyApp(programID)  
) engine=InnoDB;

create table assignmentScore (  
assignmentScoreID integer AUTO_INCREMENT not null,  
score integer,  
programID integer,  
studentID integer not null,  
primary key (assignmentScoreID),  
foreign key (studentID) references student(studentID),  

create table teacher_ThirdPartyApp (  
    programID integer not null,  
    credentialID integer not null,  
    primary key (programID, credentialID),  
    foreign key (programID) references thirdPartyApp(programID),  
    foreign key (credentialID) references teacherCredentials(credentialID),  
    foreign key (programID) references thirdPartyApp(programID)  
) engine=InnoDB;

create table student_class (  
    studentID integer not null,  
    classID integer not null,  
    primary key (studentID, classID),  
    foreign key (studentID) references student(studentID),  
    foreign key (classID) references class(classID)  
) engine=InnoDB;

This script would run in Workbench on Windows computer and would create all tables with specified primary and foreign keys, but when the same script was executed in PhpMyAdmin, installed on the server, it would fail to create four of the tables: table assignment, table assignmentScore, table teacher_ThirdPartyApp, and table student_class. The fix to this issue was to create these tables without foreign keys and
add keys later by using alter table command. Below is the code that creates and alters the tables.

```sql
create table test (  
    testID integer AUTO_INCREMENT not null,  
    testName varchar(20) not null,  
    testDescription varchar(100),  
    aveClassScore integer,  
    programID integer,  
    subjectID integer not null,  
    primary key (testID),  
    foreign key (subjectID) references subject(subjectID)  
) engine=InnoDB;

alter table test  
    add foreign key (programID) references ThirdPartyApp(programID);

create table assignment (  
    assignmentID integer AUTO_INCREMENT not null,  
    assignmentName varchar(20) not null,  
    assignmentDescription varchar(100),  
    aveClassScore integer,  
    programID integer,  
    primary key (assignmentID)  
) engine=InnoDB;
```
alter table assignment
add foreign key (programID) references ThirdPartyApp(programID);

CREATE TABLE assignmentScore(
    assignmentScoreID INTEGER AUTO_INCREMENT NOT NULL ,
    score INTEGER,
    programID INTEGER,
    studentID INTEGER NOT NULL ,
    PRIMARY KEY ( assignmentScoreID ),
    FOREIGN KEY ( studentID ) REFERENCES student( studentID )
) ENGINE = INNODB

alter table assignmentScore
add foreign key (programID) references ThirdPartyApp(programID);

CREATE TABLE teacher_ThirdPartyApp(
    programID INTEGER NOT NULL ,
    credentialID INTEGER NOT NULL ,
    PRIMARY KEY ( programID, credentialID ),
    FOREIGN KEY ( credentialID ) REFERENCES teacherCredentials( credentialID )) ENGINE = INNODB

alter table teacher_ThirdPartyApp
add foreign key (programID) references ThirdPartyApp(programID);
CREATE TABLE student_class(
    studentID INTEGER NOT NULL,
    classID INTEGER NOT NULL,
    PRIMARY KEY (studentID, classID),
    FOREIGN KEY (studentID) REFERENCES student(studentID),
    FOREIGN KEY (classID) REFERENCES class(classID)
) ENGINE = INNODB

The script above would run without errors and create a database schema for our web application.

After moving into front and back end implementation with Laravel framework our team switched from manually creating a database schema to using database migration feature offered by Laravel. Laravel’s database migration process is described in Concept of Operations section of this report.
Web Application Block Diagram

Figure 6. Web application block diagram.
Use case Diagram: Teacher’s Dashboard
Use case Diagram: Student’s Dashboard
Our Current System

Our Tools

LAMP stack

- Linux (Ubuntu Server 14.04.4)
- Apache
- MySQL
- PHP 5

Frameworks

- Laravel and AngularJS
- Front-end
  - HTML/CSS
  - AngularJS
- Back-end
  - Pure PHP
  - Laravel

Other

- Python (web scraper)
- Composer
Our Current System Development steps

First Teacher Dashboard Page

Initial teacher’s page design is shown in Figure 8. It was only a temporary prototype that gave our team some ideas on the future teacher’s dashboard.

Figure 8. Original design of teacher's dashboard.
Drawing possible design on paper (Figure 9) turned out to be a better brainstorming technique since it takes less time and still gives a good idea about the desired future layout and functionality.
After finishing paper sketches and moving into implementation, the current design for both dashboards is shown in Figures 10 and 11.

Figure 10. Teacher’s dashboard prototype.

Figure 11. Student’s dashboard prototype.
Login Page

![Login Page]

Figure 7. Color scheme and logo of the future system.
Early Project - PHP

As was previously stated, our initial project was written in pure PHP/HTML/CSS, and the above image examples are evidence of that. This section will be used to display our initial PHP code, to be contrasted with the section following this one which will contain Laravel code. Things to look for are

- The difference in modularity in the Laravel code
- How much easier to read the Laravel code
- How much more secure the sign-in form in Laravel is (of course, the security of the PHP version was just an early draft, so the insecurity would’ve been fixed had we continued to use pure PHP instead of opting for Laravel)
Login.php

Initial bare-bones login system code with php, html, and some javascript:

```php
<?php require_once("functions.php"); ?>

<?php

// check to see if teacher's name is in database

$username = $_POST["uname"];
$passwordHash = md5($_POST["pw"]);
$loginStat = checkLogin($username,$passwordHash);

if (!$loginStat) {
    header("Location: login.php?just=yes");
}
else if ($loginStat == "admin") {
    header("Location: sb_admin/index.html");
}
else {
    // $loginStat == "user"
    header("Location: user.php");
}

?>

<script>

function incorrectLoginPopup() {
    alert("Incorrect Login");

```
function unrecognizedLoginPopup() {
    alert("Unrecognized Username");
}

The script sections contains a couple of javascript alerts to inform users of incorrect-logins and unrecognized usernames.

Index.php
Simple PHP+HTML index page, tied to the login controller, login.php (shown above)

<html>
<head>
    <link rel="stylesheet" href="stylesheets/index.css">
    <title>UCP Login</title>
</head>
<body>
    <p class="header float="left">
        <img src="UCP_logo.png" alt="logo" width="300" height="121">
    </p>
    <div class="JUST">
        <p class="login">
            Login
        </p>
    </div>
<form action="login.php" method="post">
<p class="loginIn">
    username: <input type="text" name="uname" class="textBoxes" required="required">
    <br>
    password: <input type="password" name="pw" class="textBoxes2" required="required">
    <br>
    <input type="submit">
</p>
</form>
<p class="forgotten">
    forgot UN/PW?
</p>
</div>
</body>
</html>

And the associated CSS

body
{
    background-color: #58989E;
    margin: 0px;
}

div.JUST
{
    width:900px; margin-top: 2%; margin-right: auto;;
margin-left: auto; margin-bottom: auto;
border: 1px solid #FFFFFF;
background-color: #FFFFFF;
}

p.header
{
width: 100%; margin-top: auto; margin-left: 0px;
margin-bottom: 20px; border: 30px solid #FFFFFF;
background-color: #FFFFFF;
}

p.login
{
margin-top: 100px; margin-right: 150px; margin-left: 80px;
margin-bottom: 50px; width: 1000px; font-family: sans-serif;
font-size: 28px;
}

p.loginIn
{
display: block; margin-bottom: 30px; margin-right: 150px;
margin-left: 80px; width: 1000px; font-family: sans-serif;
font-size: 14px;
}
BadLogin.php

If you look closely, you'll see that this page is identical to index.php except in the part highlighted. This is when the blade templates of Laravel would come in handy.

<html>
<head>
<link rel="stylesheet" href="stylesheets/badLogin.css">
<title>UCP Login</title>
</head>
<body>
<p class="header">


<p class="login"> Login </p>

<p class="badLogin"> Incorrect username and/or password </p>

<form action="login.php" method="post">

<p class="loginIn">

username:  <input type="text" name="uname" class="textBoxes" required="required">

<br><br>

password:  <input type="password" name="pw" class="textBoxes2" required="required">

<br><br>

<input type="submit">

</p> </form>

<p class="forgotten">

forgot UN/PW? 

</p>

</div></body>

</html>
And the associated CSS

body
{
  background-color: #58989E; margin: 0px;
}
div.JUST
{
  width:70%; margin-top: 2%; margin-right: auto;
  margin-left: auto; margin-bottom: auto; border: 1px solid #FFFFFF;
  background-color: #FFFFFF;
}
p.header
{
  margin-top: auto; margin-left: 0px; /*margin-left: 230px;
  margin-bottom: 20px; border: 30px solid #FFFFFF;
  background-color: #FFFFFF;
}

p.login
{
  margin-top: 100px; margin-right: 150px; margin-left: 80px;
  margin-bottom: 50px; width: 1000px; font-family: sans-serif;
  font-size: 28px;
}

p.badLogin
{
  margin-right: 150px; margin-left: 80px; width: 1000px;
Functions.php

File that holds some of the functions used login.php. These functions are for checking for logins. The password hash that’s being checked is hardcoded and not tied to a database (again, early conceptual draft).

```php
<?php

function checkLogin ($uname, $pw)
{
    if ($uname == "admin")
        return (checkPassAdmin($pw));
    if ($uname == "parent")
        return (checkPassUser($pw));
}

function checkPassAdmin ($pw)
{
    if ($pw == "21232f297a57a5a743894a0e4a801fc3")
        return "admin";
    else
        return false;
}

function checkPassUser ($pw)
{
    if ($pw == "d0e45878043844ffc41aac437e86b602")
        return "user";
    else
        return false;
}
```
(Continued) Laravel

Database Migration

We started our web coding by creating a database migration. Such migration works as a version control for the database so the database schema can be easily recreated or updated. Migration class has two main methods: up() and down(). The code for creating tables will be inserted into up() method, and code for rolling back migration will go into down() method.

For a start we created a “user” table using migration mechanism. Later we will set up migration to perform a complete set up of database tables used by our application. Migration will be very useful when we transfer our project to school’s server.

The first step in generating a migration is to use artisan command “make:migration” shown in Figure 1.

Figure 12. Creating migration.
This will create a migration file inside App/database/migrations directory. Then we put our migration code for creating the “user” table:

class CreateUserTable extends Migration
{
    public function up()
    {
        Schema::create('user', function(Blueprint $table){
            $table->increments('userID');
            $table->string('email', 25);
            $table->string('username',25);
            $table->string('password');
            $table->string('first_name', 50);
            $table->string('last_name', 50);
            $table->string('remember_token');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::drop('user');
    }
}
The last step (Figure 2) is to run our migration:

![Image of command output showing migration process]

Figure 13. Executing migration.

The result of the migration is the “user” table created in our MySQL ucp_lg database (Figure 3).

![Image of MySQL Workbench showing user table]

Figure 14. User table created using migration.
Authentication and Security

One of our biggest concerns in regards to our web application is data security and prevention of unauthorized access to students’ records.

Laravel has built-in capabilities to protect a web application from cross-site request forgery CSRF attacks. Cross-site request forgeries are a type of malicious exploit whereby unauthorized commands are performed on behalf of an authenticated user (CSRF Protection). In our application we will need to eliminate the possibility of such attacks to prevent data corruption and access to students records.

CSRF tokens are automatically generated for every active user session within the Laravel application. They are included as a hidden field in forms so Middleware/VerifyCsrfToken.php will validate the request.

Figure 15. Example of sign up form with a hidden field that contains CSRF token from current session.
If we open developer console and inspect “Sign Up” button from sign-up page we will see our generated CSRF token:

Figure 16. CSRF token value displayed in Chrome’s developer console.
Form validation

The form validation will be performed for all form fields within our application. We started with implementing validation for sign up, sign in, and sign out pages. Even though Laravel has default register and login controllers and views, we wrote out custom ones, which fit better into our application.

Currently we have implemented such validation within AuthController postSignUp and postSignin methods. $this->validates method receives a $request object with all posted data and an array of form fields we are validating.

During the sign up process we are checking that all fields are required, password must have at least 6 characters, email needs a proper email format (built-in Laravel email validation) and is unique within our user table, first and last names only contain alpha symbols (letters) and dashes, username is alpha-numeric and unique (Figure 6).

Validation for sign in page is quite simple – it only needs to check two fields: username and password. Here is the code for sign in form validation:

```php
public function postSignin(Request $request)
{
    $this->validate($request, [
        'username' => 'required',
        'password' => 'required'
    ]);  
}
```
In case of wrong input validate method will throw a validation exception and user will be redirected back to signup page with error response displayed on the web page. We specify validation rules within controllers, for example sign up, sign in, and sign out logic is handled by AuthController. This controller handles redirection requests depending on the outcome of the validation. Fields containing wrong input will be highlighted in red when page is returned back to the user (Figure 7).

Our next step was to implement a better handling of validation by returning all entered data back to the user with corresponding error messages instead of returning a blank form with errors. If user entered data in a wrong format or omitted required fields the form validation will fail and user will be redirected back to signup page with all the fields entered remaining the same, except for the password field, that will be left blank (Figure 8). The sample of code below (from signup.blade.php – our sign up view) is managing such functionality for email field in our example:
<div class="form-group{{$errors->has('email') ? ' has-error' : ''}}">

    <label for="email" class="control-label">Your email address</label>
    <input type="text" name="email" class="form-control" id="email" value="{{ Request::old('email') ?: ''}}">

    @if ($errors->has('email'))
        <span class="help-block">{{ $errors->first('email') }}</span>
    @endif

@endif
</div>
Figure 18. Displaying errors on the sign up form.

Figure 19. Sign up page returned with filled in data and validation errors.
To create a user, we used a User Model with fields that will be filled during sign up process. Our user model description is saved in User.php

Figure 20. User model.
Below is an example of the user record created during the sign up process. For security reasons we encrypt password before saving them in a database. We use Laravel’s helper function “bcrypt” to securely hash user login passwords.

Code example (AuthController.php):

```
User::create([
    'email' => $request->input('email'),
    'username' => $request->input('username'),
    //hashed password
    'password' => bcrypt($request->input('password')),
    'first_name' => $request->input('first_name'),
    'last_name' => $request->input('last_name'),
]);
```
After validation passes successfully user authentication is handled by Laravel's Auth class. Auth::attempt() will check entered username and password against the hashed password and username stored in our database. Below is our code example from AuthController.php that checks whether authentication attempt returns true or false and depending on the outcome performs an appropriate redirection:

```php
public function postSignin(Request $request)
{
    $this->validate($request, [
        'username' => 'required',
        'password' => 'required'
    ]);
if (!Auth::attempt($request->only(['username', 'password']), $request->has('remember')))
{
    return redirect()->back()->with('info', 'Sign in failed. Please check your username and password.');
}
return redirect()->route('home')->with('info', 'You are signed in');

If user provides wrong credentials our AuthController will redirect the user back with error message (Figure 11).

Figure 22. User is redirected back to sign in page after entering wrong username/password credentials.
Explicit Design Summary

Concept of Operations

Other Systems

Our clients are afflicted with but one major problem: there doesn't exist a single unified interface in which all the teachers in the UCP Bailes school network are able to interface with each other with. Our in-progress system is the proposed solution to this. The teachers are currently using over 30 different websites and integrations, the real issue with this is that most of the websites are not standardized, so there ends up with several integrations for math, several for science etc. The problem that this poses for us is that we have to decide which integrations to support between all the different integrations competing for the same subject. Or we can choose to support all 30+ integrations and find a way to ensure interoperability between our system and the several dozen websites we may choose to support. So far, we've decided to support the following websites:

- Accelerated Reader (reading)
- Go Math (math)
- Fusion Science (science)
- DreamBox (math)
- Spelling City (spelling)

Not mentioned is google drive, and that’s one of the most important ones to support in some way as it is the primary method that the teachers currently use to share links between one another. There are plans of possibly integrating google drive into our system to act as the dropbox-like system that we need. We would of course, add the capability for teachers to comment on the links and to view comments left by other teachers.
The Proposed System: Needs

The strength of our system lies in its simplicity and its integration ability. An alternative way to build the system might've been to "reinvent" the teachers' five most-used websites (listed above) into one big mega-website. Of course, there are several problems with this idea. Not only would this be incredibly difficult and expensive to implement, it would probably not do the job as well as the different websites do their particular job. This encapsulation of goals (each website doing its own task and nothing else) resembles the UNIX way, and so too does our system. When one looks at it with that in mind, the tasks of our system become simple:

1. connect with the websites we've decided are simply necessary
2. have an easy way for teachers to and view and share links amongst each other
3. have all the integrations be viewable from one easy-to-use interface.
The Proposed System: Account Types and Permissions

For our system, there will only be three classes of users and several functions and read/write permissions that each class will exercise

- Parent/Student account: we decided that it wasn't necessary to separate this single class into two (parent and student) because there simply isn't a need for the parents and students to have different accounts. Actually that isn't wholly true, I can think of but one case in which it might be beneficial for the student and parent accounts to be separated into two, and that is under the rare circumstance where a parent or parents have multiple children attending UCP Bailes. Our current system would require the parent to have multiple accounts, one for each child in this case. So in this way, the account is really just a student account, but the details of the account is more valuable to a parent than to a student, so for all intents and purposes, this account should be designed with parents in mind, not students.

  ○ Permissions: full read for self; this means that the parent will be able to view all of their student's grades, attendance, and other information that is tied to their student's account. However, the parent/student account will not be able to write or modify any of their student's information, and obviously, not for any other student either (since they won't even have access to any other account other than their own student).

- Teacher account: The teacher account was the account that we designed first, so we ended up basing the permissions of the other account types after this account type. This was because this system was first and foremost, a system to help teachers with their organization of student data, so a lot of our initial design was centered around this idea. In terms of hierarchy, the teacher accounts rank just above parent/student account, but still below the administrator and super-administrator accounts.
○ Permissions: teachers will have full read and full write for their classrooms and all the students in their classrooms. This doesn't include write permission for the data that will be gathered externally by our web scrapers. This information isn't ever meant to be modified, so we don't plan on supporting that functionality. So since most of the student's displayed information will be from external sources, most of the modification of student fields will be changes in attendance. The teachers will of course have full read of the data gathered from external sources of course, and that permission will apply for all the students in any and all of the classes they teach.

- Administrator account: This account type has the most privileges of the deployment accounts (the super-administrator is a development account, not a deployment account). This account type is only intended to be used by the principal, the vice-principal, and other possible high-ranking officials at the school. Perhaps dean if the school employs such a position, but this account is certainly not to be used by teachers. With that said, this account doesn't have any privileges that it doesn't need. For security's sake, this account's permissions are actually surprisingly limited.

○ Permissions: in terms of read permissions, this account type will simply be an upgraded teacher account. Just like teachers are able to read all the information of all the students in all their classrooms, administrators are able to read all the information of all the students in the entire school. Although they will have school-wide read permissions, administrator accounts will have no write permissions. The reasoning behind that is simple: an administrator is not a teacher, so an administrator has no business changing a student's grades and/or attendance.

- Super-administrator account: This is the ultimate account, comparable to the root account linux system hierarchies. Such an account is never intended to be used by anyone that plans to
interface with the system, and is mostly meant for system-
development, bug-testing, and administrating behind the scenes. This account is necessary because we don't ever want to run into insufficient permission errors with any of the information that may fall into our hands, or the hands of our successors (the IT guys responsible for maintaining the system after us).

○ Permissions: the super-admin will have the exact same read permissions as the administrator (able to read all the information of all the students in the entire school), but unlike the administrator account, the super-administrator account will be able to also write to all the accounts to which it has read access to. So in short, the super-administrator account is able to read all the information of all the students in the entire school as well as write to and/or replace all the information of all the students in the entire school.

The Proposed System: Implementation

Our system will be developed first for the Linux environment. Specifically, we will be using the LAMP stack for our deployment (Linux/Apache/MySQL/PHP), and the good news about our project is that the tools we use are cross-platform, at least on the presentation side. An HTML page will essentially look the same, regardless of whether or not it's being displayed on Linux, OS X, or Windows, assuming the same web browser is being used on all three platforms (even if different browsers are used this will be the case, except in some niche instances where a browser renders and displays the elements differently than from other browsers).
Project Plan

Process

The process of creating the software will include; building, prototyping, testing, and evaluation. Building and prototyping intentionally overlap, as does testing and evaluation. The first phase, building, consists of designing and laying the foundation to hold the system up, as well as instructions on how to set up the system initially. Prototyping, phase number two, occurs as soon as the database and framework are stable and can accept basic requests. The third phase, testing, includes the different types of testing to occur, as well as details on how the sponsor and development team will interact. The final phase, evaluation, will conclude the project and involve heavy sponsor feedback and reflection.

Please refer to the Milestones section for an overview of these phases and the dates at which they are set to begin.

Building

Design Principles

Creating the software follows the design principles created from initial planning. The design principles are established from the goals, specifications, and requirements that are explained in detail under the Project Description and Detailed Design Content sections of this document.

The system can be broken down into three main components; the server (“back-end”), the database and overlying framework, and the user interface (“front-end”). With these naturally occurring divisions, the team decided that similar divisions in the allotment of tasks would be appropriate to build the system, based on the experience, specialties, and interests of each team member.
Project Management & Scripting

Chris is the project manager, and is also supporting the back-end with scripting. Chris has had internship experience and is familiar with testing procedures, methods, and industry code. Chris will also be the group member primarily interacting with the sponsors and obtaining feedback.

User Interface

Alex St. Louis will head the development of the user interface, working closest to Lyudmila. Though the whole team has visibility of the entire project, Alex will be most involved with the design, features, and organization of the user dashboard.

Database & Framework

Lyudmila is leading the development effort of the framework and database, and has experience from coursework and an internship to reinforce a responsible and logical approach to this system. A modern framework and well-known database are to be used to keep the system robust for the foreseeable future. Managing requests to the database and integration with the user-end of the website are a couple of the more general tasks required.

Server

In charge of the server and all related tasks is Mario, who has had previous experience with the Ubuntu-server distribution of Linux. In setting up and maintaining the server, it is necessary to install the programs needed by other team members to develop, manage user and account settings, perform essential updates if needed, and otherwise ensure that the security and permissions of the system are maintained.

Supportive Roles

Despite primary roles, all group members will continue to support and interact with each other, taking on tasks outside of their primary area of
focus. For example, other group members usually accompany Chris to meetings with UCP Bailes, and the entire group is becoming familiar with the framework of choice to assist with the effort to build a robust system. The intention is to ensure that all group members are familiar with the entire system, and to foster a cohesion between our respective tasks that works well--and naturally so--rather than forcing and hacking the individual components together.

**Development**

The overall general flow of development works from the back-end to the front-end. Server and database management must be in place for core operations, and then the front-end can build on the foundation of the back-end. With the proper tools and setup, it should be possible to add additional functionality (such as “stretch goals” that have already been identified, and future suggestions by the sponsor) between iterations of the prototype and shape the system so that core operations remain stable. The interface should remain malleable and open to additional implementation; the goal is to build a general structure that has enough built-in leeway to be adopted and customized by other projects, and then build off of this base structure to fit the needs of UCP Bailes elementary.

Developing the system is done incrementally with weekly goals and meetings. Meetings outline the current progress, changes in plan, and goals to be accomplished in the following week for each team member. Following the status update, if time permits, the team will work on small project tasks and critique current work. This provides the opportunity to adapt as the project develops, loosely following the agile methodology.

**Tools**

Tools for communication and development include Slack for communication and git (using the GitHub online service) for version control. Slack is a messaging application intended for team communication. It is available for both mobile devices and personal computers, and allows for integration with other productive applications such as Trello. GitHub is an online repository that makes all source code available through a URL and also provides a wiki for project
documentation. GitHub is a popular choice among open-source projects due to both its online development tools and free public repositories.

Each major step of building will be documented, and guidance for particularly difficult sections of development will be maintained on the project wiki (attached to GitHub) to allow future developers to recreate their own system. The documentation will also provide a failsafe for the team to ‘reboot’ the project if drastic or otherwise tragic unforeseeable events occur, and all progress is lost.

Testing

Testing will be conducted continuously, as parts are formed and integrated, to reduce the risk of developing an unstable system. Specifically, the following testing is expected:

- Unit testing
- Integration testing
- System testing

Testing will be performed by both the development team and by an experimental group of instructors within UCP Bailes during the fall semester. Testing within UCP Bailes cannot occur over the summer due to the recreational focus of the summer camps that the school offers the student body in the summer.

Unit Testing

Unit testing will occur throughout development, by the development team. It will primarily consist of simple pass or fail test cases and plausible edge cases. Testing will occur with false “dummy” data of small scale, and then scale up to larger data as the team acquires the data that typically accumulates from a full class at the school.
Purpose

The purpose of unit testing is to verify that smaller components properly execute before adding them onto the larger system. In this light, the goal is to produce reusable components that can be integrated with the other components, as needed. By isolating each portion of the system, in both development and testing, we hope to produce compatible components that fit together well, and can be modified as needed with as little overhead and configuration between components as possible. The goal is to allow other developers to acquire and modify the source with as little overhead as possible.

Front-End

Team members will manually develop and execute the tests for the front-end, as the user-interface is expected to constantly evolve with the feedback from the end users.

Back-End

Team members will manually develop and execute tests for the back-end, though automated testing is a possibility as there is only interaction with the database and back-end scripts. Handling only data is simpler to automate through bash or python scripting, and has the added benefit of consistency and quick execution relative to manual testing.

System Scripts

System scripts include, but are not limited to, web-scraping scripts, CSV-file parsing and creation, data processing, and grade evaluation scripts. Team members, initially, will manually develop and execute tests for system scripts. After prototyping confirms that primary functionality is stable and that design is unlikely to change, automated testing may be implemented to run for every following prototype release or major addition to functionality. This last goal overlaps with integration testing.
Integration Testing

Integration Testing will occur primarily in the middle portion of development, when the project completion is estimated at twenty-five to seventy-five percent. Integration testing is anticipated to begin around the first prototype, and continue heavily over the summer and lead into the beginning of the fall semester, when we expect to install a real-time system at UCP Bailes to handle the full load of real student data from select classrooms.

Testing will be more extensive than unit testing, as it will focus on how the parts--already subjected to unit testing at this point--interact and handle a variety of data. For example, creating an account, navigating to the dashboard, clicking a button to parse and upload data from a CSV, clicking a button to export data to CSV, and so on. Similar to unit testing, plausible edge cases will also be tested, such as attempting to export data to a CSV when the database does not contain the requested data.

Purpose

Integration Testing is evaluating the performance of the units with respect to this project. It’s a means to improve efficiency and tailor the interaction among components to provide a reasonable response time to user queries and appropriate data formatting and input along with output.

Front-End

The primary concern for the front end is that buttons and navigation perform the requested action, and successfully initiate events in the back-end to generate proper results from the initial query. Testing will essentially be performed manually, as the user-interface is expected to constantly evolve with the feedback from the end users.

Back-End

Team members will manually develop and execute tests for the back-end, though automation of testing and verification may be implemented. An example of back-end testing could be parsing an existing CSV and
entering it into the database, which would be the same event that would occur when the user presses the appropriate button on the front-end.

**System Testing**

System testing will occur at the very end of the project, in late November or early December.

The system as a whole will be evaluated in accordance with the requirements specified within this document. Testing will first be performed by the team, to ensure that requirements of the system are met, that any stretch-goals that were implemented were properly achieved, and that the system is reproducible on a clean install of an Ubuntu server to allow for reproduction in the future if the school expands or the system is adopted by the other UCP-associated establishments in the area.

System testing encompasses the experimental group of teachers who will have the system in their classroom during the fall semester (see Prototyping). The subtle faults and failures in the system are expected to surface while the system has a full load for an extended period of time. If captured, attempts will be made to remedy faults if they impair the performance of the system.

**Feedback**

Although the primary method of communication is through email, additional steps should be taken to ensure prompt and helpful feedback from the users while the system is in use throughout the prototyping and testing phases.

The ideal solution is to create an online form, accessible from the dashboard, that allows some measurement of user satisfaction (e.g. a Likert scale) and a free-response textbox for suggestions, critiques, and bug reports.
Evaluation

Evaluation of the software will rely on the feedback survey noted under the Testing section of this document, as it is the most practical given the schedule conflicts between the hours of UCP Bailes and the daily schedule of the development team. However, with each major feature of the system, or at least every eight weeks, the development team would like to arrange a physical meeting with UCP Bailes for demonstration and collective feedback.
Related Work

The problem of storing, managing, and displaying a collection of data is by no means a new problem. It is not even a unique problem to develop a system to store, manage, and display data for a school system. In fact, their are a very wide variety of learning management systems (LMS). These systems have been built for both academic and corporate use, and boast a wide variety of features. The features that these systems implement include creating and distributing files, lesson planning, and web based courses. Here is a list of a few popular learning management systems:

Popular Learning Management Systems

- Bridge
- Blackboard
- Canvas
- EduWave
- SEQTA Software

These tools are very great for administering assignments and tests as well as providing online training. In fact, our application is not meant to replace all forms of alternative LMS entirely. Those solutions have been developed with a very large team over a more extensive time period, so they accomplish tasks that are out of the scope of this application. These tools may solve a few problems, such as interfacing directly with the students for tests and assignments, and solve them well, but they do not directly cater to the specific needs of UCP. Here are some common features among learning management systems that our application will attempt to replace or integrate with:
Common LMS Features

- Classroom management
- Gamification
- Gradebook
- Mobile learning
- Student portal
- Testing/Assessments
- Blended learning (combining online with face-to-face)
- Reporting
- Surveys
- Permissions
- Groups
- File distribution
- Email integration

The problem with the “popular learning management systems” mentioned above is that they are way too expensive for UCP. These products offer a single application to rule them all. In other words, the learning management systems listed are designed to replace every other tool that a teacher might want to use. This would include things like administering tests, creating assignments, storing grades, and reporting student progress. Since UCP cannot afford such a solution, they use a wide array of smaller tools.

Each tool is designed for a specific purpose, and it takes all of the tools to manage a class on a day-to-day basis. For example, some tools may be used for creating math assignments, while some may be used for only spelling. This makes the teaching process slightly more complicated, but
can work well if managed properly. The problem comes when the teacher, parent, or administrator would like to see the results gathered from these tools. It is a very convoluted and frustrating process to navigate many applications, file systems, and web sites just to figure out how a student is performing in a given subject or on a given assignment, let alone to decide the overall performance of a student across all of their academic involvements.

Instead of attempting to phase out or replace all of the current LMS tools that the teachers use and are comfortable with, this application will integrate with those systems, using their data to provide for the specific requests of the UCP Bailes administration. Here is a list of applications and platforms currently used by the UCP teachers and administration to manage their students:

Currently Used Solutions

- **Dreambox**
  - Used for math subjects
  - This application implements the gamification, as all tests and assignments are designed to be fun and rewarding
  - Dreambox also integrates with each student’s Ipad, giving students a blended learning

- **Spelling city**
  - Used to give spelling tests and exercises
  - This application gathers a large amount of data, as a student will regularly spell many words, and each word is a statistic

- **Think Central**
  - Used for math subjects
  - This application was regarded by the teachers to have a very bad interface
In order to complete a given task, there is a series of steps that you go through, without instruction or feedback from the site, in a specific order, while navigating across different sections and tabs in the application.

This application will be used as inspiration for what not to do.

- **Google Classroom**
  - Used for any subject to administer a custom test or distribute an assignment.
  - Very easy and intuitive interface, but very limited functionality for UCP teachers.

- **Accelerated Reader**
  - Used for reading.
  - Simple user interface, but limited functionality.

- **Gateway**
  - Used primarily as a gradebook for teachers to enter grades manually.
  - This is what our application is intending to replace by adding much more automated features to aid the teacher.

- **Google docs**
  - This is the primary method for distributing files, such as lesson plans, behavior forms, and student performance data.
  - Google docs is used for passing documents between teachers, administrators, and even students, as there is no unified network to communicate across.
  - To integrate our application with google docs, it will support the posting/sharing google docs links with a specified target audience, as well as adding the functionality to comment on these posts.
This is a small list of the many applications that are used by the teachers at UCP to manage their day-to-day work involving the students. Their primary challenge is keeping track of all of these different applications and gathering the data they provide. The other challenge is sifting through all of the data provided across all of the various platforms in order to pick out the most important data.

Our application will gather data from all of these various platforms to store it in a single place that is easily accessible by anyone with the valid permissions. It can then display the most important student data to accurately portray the progress and performance of that student. It would be ideal to display this overall student summary as compressed as possible. That way, with a quick search, or a few quick clicks, the teacher, parent, and administrator can easily see how the student is doing in all aspects of their schooling. Then, if the client would like to get a more in-depth look at the student progress, our application can be navigated to gather more detailed data from the various sources.

Our system will use the current popular LMS tools that are out of the price range of UCP as a basis for our design. The team can learn from what these LMS tools do, and make note of the features that are feasible for our project. Some aspects of those applications, such as design, can be used as direct inspiration for this application.

This group also needs to focus, maybe even more so, on the tools that are currently being used by the UCP teachers. Those tools have tested extensively by the teachers, and it will be clear to determine their good qualities and their bad qualities. Based on this teacher feedback, the team can successfully devise a system that functions to their specific requests, as well as integrates with those current tools, so that they do not have to complete adopt a whole new method of teaching. Using the issues of the popular LMS tools as well as the issues of the currently implemented LMS tools, our group can design a system with the teacher’s interests in mind. Here are some of the major issues with both of those sets of LMS solutions:
Issues with Current Solutions

- Cost
- No integration with technology-based classrooms
- Unintuitive interface
- Not well suited for students with disabilities where performance is solely based on a letter grade
- Does not generate reports or visuals
- Not easy to communicate with parents
- Very specialized for a single subject

Cost

This system is going to focus primarily on satisfying these key issues, as that is what the this school specifically needs and will set us apart from the available options. The first point is by far the most important point. UCP is a nonprofit charter school, meaning that they do not have a lot of money to spend on this system. Our system will be by far the cheapest option available because it will provide all of these features for no additional charge. The only cost will be the cost of hosting the server to store the data. However, according to the UCP administration, they may be able to use a their own server, which would essentially cut the cost to be only the maintenance of the server and this application.

No integration with technology-based classrooms

The next point, integrating with technology-based classrooms is also very important. At UCP Bailes campus, they implement a 1 to 1 IPad initiative. This means that in some classrooms, every student has their own IPad to aid in the learning process. “Now all of our Bailes’ K-5th grade students have their own iPads for academic school work, providing them with an effective educational tool that’s already improving their learning potential [3].” It is clear that if an LMS does not offer support for technology
integration, it will be far less effective in the classroom. Our application will be sure to integrate well with the current solutions that implement these techniques. For example, Dreambox allows the students to use an iPad app to complete assignments and tests. This data will be sent to the Dreambox server, and eventually to their web portal. From there, it will be possible to grab that data using various possible extracting techniques, and display it in our application.

Unintuitive interface

After a few meetings with both the administration of UCP as well as a UCP teacher, our team realized how important the user interface is. Many of these teachers are not as fluent with new technologies, rendering an already complicated interface unworkable. The interface must be extremely simple, displaying only what the user needs to see. Each type of user (Administrator, Teacher, and Parent) will have their own user interface, that molds the display to their specific needs. It is also very important not to bombard the user with information when all they want is a brief overview of a particular student’s or class’s performance. For this reason, the application will only display the most important information to the user when they first navigate to a page. Then, the user will have the option to expand that information later.
Not well suited for students with disabilities where performance is solely based on a letter grade

One of the biggest challenges this team will face in the development of the application is how to accurately display a summary of student performance. In meetings with the sponsor, they emphasize that the students at UCP do not always fall into the traditional letter grade system. In other words, you cannot just look at a student’s raw test scores, find the average, and accurately label the performance of the student. Most of the student performance is evaluated based on progress, and is somewhat scaled based on the level of disability a particular student has. It will be very important to emphasize in our application the amount a student has progressed throughout the year. Based on the school subject, there may be specific rules for what goes into the progress of a student. For example, for a spelling oriented subject, the total number of words spelled correctly would be enough to evaluate student progress. However, in a math oriented subject, it may be more important to focus on the mastery of a given topic, or at least the progress on a given topic, rather than just the change in the student’s grades.

Does not generate reports or visuals

It is absolutely essential that our application implement some form of report generation. The teachers at UCP must present progress reports at meetings on a very regular basis, possibly as much as once a month. The current approach to this problem is creating an excel spreadsheet, which will automatically generate a somewhat decent looking graph once all of the data is entered. However, the only means these teachers have to grab all of the necessary data is a manual entry. This can take up hours of a teacher’s time. Once our application is properly integrated with all of their tools, and it has all of the data needed, the application will provide them with various graphs and reports.
Not easy to communicate with parents

This day and age, communicate with parents is vital. All parents want the best for their children and are very concerned with the performance of their student. The current system is not set up for easy access of the data relevant to students. If a parent is asking about student evaluation, the parent will just have to respond via e-mail, text, or other various messaging services. This means that the teacher will have to visit many different platforms, write down all of the data found, then formulate a message based on this data. All of this interaction is done case-by-case and requires delicate attention. Once this application has gathered all of the student performance data, the parents will be able to use their account, linked to a single student, to easily view all data corresponding to that student. This also supports real-time updates for this data, with very little user interactions needed to gather and display the data.

Very specialized for a single subject

The tools that are currently being used by UCP teachers are by no means scaled the same way that the more expensive LMS tools are. Each of these tools used will typically only provide enough functionality to support a single school subject. This means that in order to be able to use a single platform, with all student data stored and retrievable on it, the school would have to spend much more money. However, with our application, since it will integrate with many of current tools they used, they will be able to have all of their data in a centralized and structured in a single location.
Facilities and Equipment

Development

The University of Central Florida has provided a virtual machine, Ubuntu Server 14.04.4 (“Trusty Tahr”), on the Senior Design Lab server. The virtual machine has two processors, four gigabytes of RAM, and thirty gigabytes of storage.

While the UCF server is adequate for development and initial prototyping, loading real student data would lead to security concerns and possible complications with UCF-affiliated network policies. The development builds are for use of the development team, and a separate server should be acquired for builds that require real-time student data and loads.

Implementation

Most online data-hosting services have been discounted at the time of writing. There is the prospect that UCP Bailes will acquire funding to purchase a low to mid-range private server, or claim server space at the central UCP technology department located in downtown Orlando.

Once a secure server space has been established and the prototype is stable enough, a working build of the system will be replicated on the UCP Bailes private server, at which point the UCF server will be maintained for unit and integration testing.

There is safety in handing off the responsibility of private server maintenance to the UCP technology server, as they may have standards in place for ensuring security and longevity of data. If the UCP technology center is unable to provide a proper server, then the development team will need to responsibly suggest an able server to handle the needs and growth of the Bailes campus.
Sponsor Interaction

Sponsor

The target user, UCP Bailes, can be considered the sponsor of this project. There is no monetary gain or compensation aside from the personal fulfillment of assisting a nonprofit, the learning experience, and achieving the graduation requirements for the Computer Science undergraduate program.

Primarily, the development team will be interacting with a select group of teachers and the administrators. Teachers will be questioned for general requests of and comfort with the system, while the administrators will be contacted largely for logistics of implementation and the overall structure, permissions, and law (privacy or otherwise) concerning the completed system.

Identifying Sponsor Concerns

Initial planning with the sponsors revealed that only one or two staff under the Bailes umbrella have some understanding of the technical details behind the system, but are unable to program. Understandably, the other staff are only familiar with the websites and online tools that they currently use, such as Google Drive. Therefore it is up to the development team to identify the needs, requirements, and feasibility of the project.

Purpose of Physical Meeting

Physically meeting with the sponsor is preferred when speech is more convenient than text. In this context, “sponsor” refers to the collective group of the educators and administrators primarily involved with the project. In order to avoid misinterpretations, notetaking and multiple team-members present at each meeting is preferred, since much more information can be compacted into a brief meeting, and details can be lost in the pace. Physical meetings are more convenient for the sponsors when located at UCP Bailes, and should occur promptly around the following events:
- Critical feedback or changes to the layout is requested by the sponsor
- A major design flaw is identified
- Visual or interactive functionality requires sponsor input
- Questions about obtaining student data from websites, in detail
- Implementation and training for the software
  - Prototypes
  - Completed build

For each meeting, the following should be identified, so that the team can design and develop with better accuracy for future iterations:

- What the sponsor liked
- What the sponsor did not like
- The ideal solution from the sponsor
- Artwork or sketches, if necessary
- Additional guidance or direction, such as a live website that can be evaluated as an example
- Potential trade-offs and scope-impact that new functionality will incur on the limited development schedule
- General action items for each team member

The team should then evaluate how to best approach the new problem, if changes to the current design model is necessary, and update milestones accordingly to reflect the conclusions of the meeting.

Email Communication

Email will be used for simpler questions, most documents, non-sensitive data and lists, and online resources that UCP Bailes feels is of importance. Email communication is preferred for non-visual content and website URLs, as it is persistent and usually carbon-copied to all team members. Text-based communication has an advantage over physical meetings, where there is a higher potential to introduce error into design by relying on human memory and quick notes.
Additional Communication

As mentioned under the Testing and Evaluation sections of this document, online survey or feedback forms have been suggested for use in the future. The form would be placed on the user-dashboard and would allow the teachers who may not have team contact information to report bugs or suggestions.

Sponsor Contacts

The sponsor contacts are listed below. More contacts may accumulate with development.

Administration

- School Administrator: Anna O'Connor-Morin
- Assistant School Administrator: Thomas Brickle

Note: Tom has only been with UCP Bailes for about a year, and so is not as familiar with the day-to-day problems of the current system as the classroom educators. Tom does, however, offer great insight into administrative tasks that could be incorporated into the dashboard.

Classroom Educators

- First-Grade Educator: Doris Lawson
  - Primary contact regarding system design and user front-end
- Educator & Technology Specialist: Brett Weiss

In our first meeting with the sponsor, we talked with Tom Brickel, the assistant principal
Suppliers and Consultants

Supplier

The University of Central Florida is supplying the server and virtual machine on which the initial development and prototyping will take place. The server is a part of the Senior Design lab. The make and model of the hardware is not known as of this writing, but the system has been allocated 2 processors and 4 gigabytes of RAM for an Ubuntu LTS 14.04.4 Server virtual machine.

Consultants

The advisor and consultant of this project is Dr. Mark Heinrich. Dr. Heinrich has insight into the development and scope of the project. Any significant changes to scope shall first be evaluated by Dr. Heinrich. If Dr. Heinrich approves, only then will changes be made to the plan of development and system design.

Mr. Michael Powell will also be in contact with the team and consulted for advice or issues surrounding the UCF server and transition to a private server. Mr. Powell has access to the server, in case there is a severe mishap and the virtual machine needs to be pulled down.
Administrative Content

Budget
Due to the fact that our sponsor, UCP Bailes, is a non profit charter school, there is essentially no budget for this project. The server that will be used for development and testing will be provided by UCF. Once the application is complete, it will transition to a server provided by UCP. According to the administration at UCP, there is a possibility that this application can run on a server that they already own. If that is not the case, they will be responsible for finding a server to run the web application on.

Milestones

<table>
<thead>
<tr>
<th>Date</th>
<th>Milestones</th>
</tr>
</thead>
<tbody>
<tr>
<td>03/03</td>
<td>Research 10 software programs used by the teachers</td>
</tr>
<tr>
<td>03/04</td>
<td>Deadline for deciding on which type of DBMS/server bundle that we will be using</td>
</tr>
<tr>
<td>03/05</td>
<td>Set up server</td>
</tr>
<tr>
<td>03/08</td>
<td>Assign roles to group members</td>
</tr>
<tr>
<td>03/15</td>
<td>Crude UI mockups; images, diagrams, etc. should be drawn by this time</td>
</tr>
<tr>
<td>03/22</td>
<td>Final design of the database structure</td>
</tr>
<tr>
<td>03/30</td>
<td>First prototype of the project. Basic user interface, with connections to database</td>
</tr>
<tr>
<td>Date</td>
<td>Task</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>04/02</td>
<td>Begin working on authentication system</td>
</tr>
<tr>
<td>04/10</td>
<td>Complete authentication system</td>
</tr>
<tr>
<td>04/12</td>
<td>Have fully-functional website scrapers working for the 3 most used websites</td>
</tr>
<tr>
<td>04/28</td>
<td>Final Design Document due</td>
</tr>
<tr>
<td>05/05</td>
<td>Term ends</td>
</tr>
<tr>
<td>05/20</td>
<td>Deliver basic prototype to UCP</td>
</tr>
<tr>
<td>07/15</td>
<td>Complete all scrapers for websites without API</td>
</tr>
<tr>
<td>07/30</td>
<td>Finish Unit testing for scrapers</td>
</tr>
<tr>
<td>09/16</td>
<td>Get feedback for the prototype from the teachers and staff at UCP Bailes</td>
</tr>
<tr>
<td>10/31</td>
<td>Update initial prototype from UCP feedback</td>
</tr>
<tr>
<td>11/01</td>
<td>Final Integration testing</td>
</tr>
<tr>
<td>11/13</td>
<td>Final System testing</td>
</tr>
<tr>
<td>11/27</td>
<td>Deliver complete web application</td>
</tr>
</tbody>
</table>
Project Summary and Conclusions

The primary goal of this project is to assist the UCP teachers and administration with their day-to-day activities, so that they can focus more on achieving their goal, giving every student a chance to learn and excel. This project will be able to ease the workload of the UCP staff by managing their data. This data includes student performance metrics (assignments, tests, quizzes), student attendance, behavioral forms and various teacher files, such as lesson plans. The management of this data is extremely tedious and inefficient. This application will automate and standardize this process, making it easier for teachers, administrators, and parents to update and access this data.

The system to manage this data will be comprised of a web portal, a web server, and a backend database. The web portal will be used to display all of the data in an intuitive way that easy to read and navigate. The web server will be responsible for interacting with the web portal and updating the database. The database is where all of the gathered data shall be stored. Using these three components, the application will be able to gather the relevant data, store it, display it, and update it.

By using this system, the UCP staff be able to more efficiently communicate with the parents, other teachers in the school, and effectively the students, as they will be spending less time managing their data and more time focusing on the students. This system will ultimately assist UCP in upholding their mission statement, “To empower children with and without disabilities to achieve their potential by providing individualized support, education and therapy services in an inclusive environment [2].”
References


