University of Arkansas – CSCE Department
Capstone [II] – [Final] [Report] – Spring 2020

# Student Dismissal Application

**Ange-Thierry Ishimwe, David Rowland, Gabriel Del Carmen, Khern Toussaint, Tory Adderley, Tomas Stevens.**

## Abstract

School dismissal is a stressful activity for both parents and teachers of young students. Our application aims to streamline this process by helping teachers to dismiss students to the correct mode of transportation and notifying parents when their child has been dismissed. With this project, we intend to have students leave school safely and efficiently. The efficiency of the proposed system helps to put instructors and parents at ease and streamline the dismissal process.

## 1.0 Problem

School dismissal from elementary and middle schools is often a chaotic ordeal. Numerous safety issues need to be closely monitored by staff during this time. The teachers on duty are responsible for keeping the children away from moving vehicles while also ensuring that the car they are entering is the correct vehicle. Students riding the bus may not know the procedure or bus number they need to ride, and it's difficult for teachers to ensure student safety while also ensuring an orderly exit from the facility. Delays anywhere in the process frustrate parents and keep teachers at work longer than necessary.

While safety concerns are paramount at schools for young children, good safety protocols inevitably come at a cost to efficiency. By having an app that any teacher with a smartphone can use, we can reduce the burden of these protocols and improve the efficiency of the dismissal process. If the teacher has access to the correct mode of transportation and particular car or bus for each student at their fingertips, they can focus more on keeping children out of immediate danger and keep the lines moving. Teachers will have dynamic access to the location of each student and their current stage in the dismissal process without scanning printed roster sheets.

## 2.0 Objective

The objective of this project was to provide teachers with a tool for a more efficient school dismissal, and parents with a notification system so they can respond quickly to any errors that may occur in the process. Car lines will move more smoothly, in turn, reducing traffic and congestion around schools, children will miss their bus less often, and teachers will finish the dismissal process faster, having safely dismissed all students.

## 3.1 Key Concepts

This application is a full stack application. A full stack application consists of two main components: Client Side and Server Side. These two components of the application are designed to communicate and transfer data between one another. Each component serves a specific purpose to accomplish the overall goal, and each will be briefly discussed below.

The client-side of the application can be used by the user to read, update and delete any important data. For example, a teacher user has access to which students have left already and which type of transportation each student will take on a given day. The client-side has a graphical user interface to allow the user to easily manipulate their data. The client-side consists of both a mobile and a web component.

The server connects to our database to serve information using a RESTful API. The database is a cloud-based MongoDB instance. The API deals with requests made by the user to make various changes in the database. It is able to handle requests from multiple users. The database is responsible for storing important data and ensuring that all authorized users have easy access to whatever information they need at any time.

## 3.2 Related Work

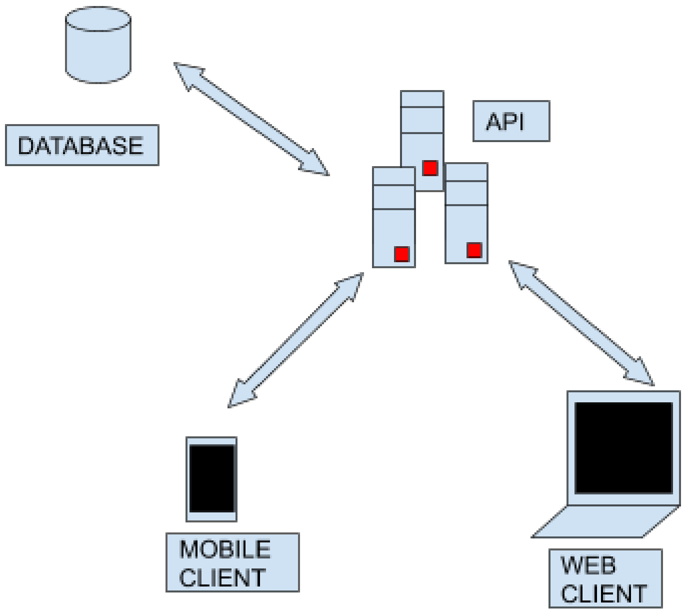
There are a few alternatives for school dismissal programs. *School Dismissal Manager*, *PikMyKid*, and *FetchKids*. All of these platforms are custom solutions that have a subscription and installation fees in the many thousands of dollars. We want to develop something cheaper the school could install themselves, by possibly having a website that an administrator could populate with student data rather than have an IT person migrate or build a database from scratch. By building an easy to use solution, teachers and administrators will have the ability to manipulate, manage and keep track of student data.

# 4.0 Design

## 4.1 Feature Requirements

1. Administrators are able to build and modify the database without any database query experience.
2. Teachers have access to their students' transportation data at any time.
3. Client applications have a rich and responsive user interface.
4. Teachers on duty will be able to fill the car-line queue by entering numbers displayed in parent vehicles.
5. A second on duty teacher will be given a continuously updating list of the students whose parents are waiting, in order, and be able to delete a student from the list when they safely enter their ride.
6. Each dismissal is recorded in the database as a dismissal object. The collection is indexed by the document timestamp for easy retrieval at a later time.
7. On duty teachers can quickly pull up a list of the students in a class based on that class' teacher.
8. Application gives notifications to parents when a student is dismissed.

## 4.2 High Level Architecture



The data persistence is handled with the NoSQL database MongoDB. The database queries are not very complex and each student has a unique number which suits the key-value structure of NoSQL. We created a RESTful API to serve the database info with the help of the Spring Boot framework. The API handles all requests from the client side. For example, one route triggers the API to send an email to the students parents if available. The API handles requests from multiple clients, and Spring Boot is built with enterprise solutions in mind so it would be able to scale to an actual production environment.

There is also a Single Page Web Application with a simple and intuitive user interface so administration can maintain and update data within their own database without constant technical support. The web application offers all the functionalities that an administrator would need to manage the database. The Single Page Application was made using the React.js library.

In addition, we built a mobile app with features to aid teachers in keeping track of a car line queue, among other things. We used the React Native framework so the same code could be used to create similar experiences on both iOS and Android. The design between the web and mobile application were consistent. Each of these four components communicate with each other reliably to aid the user in completing a variety of complicated tasks.

## 4.2 Detailed Architecture

### *Backend*

The MongoDB instance is deployed to the Atlas cloud service. It is actually supported by the same people that created MongoDB so set-up and integration was relatively simple. There is thankfully a free tier with limits well within the scope of this project. We created a JSON representation of our test data and their software easily writes such a file to any collection you would like.

The API was written in Java with the help of Spring Boot and Maven. Spring Boot is an enterprise grade dependency injection framework that is really great for setting up RESTful web services. By using a system of @ annotations and pre-built classes you can quickly build a backend by creating a small number of components they call “beans”. The framework is also “opinionated” which means that if you don’t supply a particular environment variable or component, Spring Boot takes it’s best guess. This makes it remarkably error tolerant which was a big plus in the project’s development.

Maven is a dependency management system wherein you supply an XML file with all your project’s dependencies. The Maven system then downloads any external libraries (such as Spring Boot) at compile-time to produce a functional build.

## *Website*

The website is a single page application that is written in the javascript user interface library called React.js. React allows the developer to create rich and simple web applications using JSX. React was made by Facebook to build user interfaces for front end web applications. We decided to use React to build and design the website because of it's high performance and easy to use setup.

As mentioned earlier, the website allows users to see the full list of students with all their accompanying details such as name, guardians, mode transportation, etc. When the List All Students view is selected, our website needs to communicate with our API to receive and show the most up to date list of students. To achieve this, we use the Axios library, which is a library that handles HTTP requests and makes it easier to interact with servers or other websites. Therefore, when the List All Students view is selected, the website sends a GET request via Axios to our API to fetch and display all the students in a table.

The second functionality is the Create Student option, which allows users to add a student to the database. To add a student, users must first fill in the fields such as name, class, teacher, etc., and when all the fields are filled in, and the Create Student button is pressed. Our program changes some of the inputs data-types, to match the data in our database, for example, changing the student ID from string to an integer. Lastly, all the fields are combined into one object, which is sent to the API through Axios. When adding a student is successful, the user receives a notification.

The third functionality is the Update Student, which allows users to alter any student's data in the database. To update a student's data, you need the student's ID and inputs to the fields to update. Therefore, when the update student button is pressed, the website sends a GET request to the server to fetch only the data of a student with a matching ID. Subsequently, the program only updates specified data on the student's object, which is then sent back to the database, and an alert is sent to the user that it was successful. The last functionality is the Delete Student, which allows users to delete a student from the database. To achieve this, we only need to send the student's ID to the API, which then deletes the student.

## *Mobile Application*

The mobile application was written with the React Native library. Similar to React.js, React Native is a component based UI framework for mobile applications. In React Native, each component instance has its own life cycle and state. This allows for the creation of reusable components that can be modified as rendered to suite different program needs. A big gain with using React Native is the ability to write one codebase (in Javascript or Typescript) that can provide an almost identical experience across both the Android and iOS ecosystems. React Native is also one of the most popular UI frameworks and consequently has an absolutely vast selection of pre-made open source components for use in any project.

We used the react-navigation library for our mostly tab-based UI navigation, and MobX to handle the state management. MobX is an alternative to React Redux that uses special decorators like @observable, and @action to trigger the application to listen and respond to changes to the appropriate underlying data structures. For example, the car queue feature operates by getting a list of all the objects in the Car-line collection, throwing them in an array, and sorting them by relative position number. Each screen that displays the queue is connected to a MobX data store representing the database collection, and if the contents of the collection changes then each of these components update their view. In that way we ensure that every user is seeing the exact same thing when viewing the same tab.

## 4.3 Risks

| Risk | Risk Reduction |
|---|---|
| Parents looking at phone in line | The parent app has no use in the car-line. The parent app is for notifications only |
| Secure student data | SSL for transmissions and user roles and user authentication for different levels of functionality within the app. |

**4.4 Tasks** – Below is our original task management window that we attempted to follow throughout the semester.

1. First we will do research on alternative applications, structures, and other tools that can help benefit our application in any way shape or form. We will dive deeper into the best language, algorithms and design that allows for smoother workflow of the teachers and parents alike.
2. Decide the approach and technologies that will best suit the needs of the project.
3. Design and build the database schema and structure to handle the data and provide quicker responses, read ups, and manipulations.
4. Design and build robust back end system.with key functionalities.
5. Design and build the client applications to easily communicate with the back end system.
6. Once finished developing core components, improve the quality of the overall application and optimize any parts that can be implemented better.
7. Test extensively for security flaws, breaches, crashes, bugs, and other issues as much as possible.
8. Field test at some point using some fake school area to see if full implementation was a success.
9. Write a report and presentation on the application and implementation of said project.

## 4.5 Schedule

Below is the proposed timeline we created corresponding to the tasks above. Each was estimated to be a two week period that has at least 1 person in charge with everyone supporting or implementing key features of their own.

| Tasks for two week period | Who Is In Charge | Dates for each task |
|---|---|---|
| 1. Research | Everyone | Current date -1/13/20 |
| 2. Design and Build Database | Tomas, David | 1/13/20 - 1/20/20 |
| 3. Design and Build API | David | 1/20/20 - 1/31/20 |
| 4. Design and Build Client Side (Mobile) | David, Gabriel, Thierry | 1/31/20 - 4/25/20 |
| 5. Building Client Side (Web) | Tomas,Tory, Khern | 2/14/20 - 4/25/20 |
| 6. Unit Testing | Everyone | 4/25/20-4/28/20 |
| 7. Integration Testing | Everyone | 4/25/20-4/28/20 |
| 8. Field test | Everyone | 4/25/20-4/28/20 |
| 9. Presentation | Everyone | 4/29/20 |

## 4.6

## Deliverables

1. Design Document: Contains a listing and brief description of each major component.
2. Website for teachers to modify and add to the student database.
3. Android/iOS application for teachers
4. Android/iOS service for notifications to parents

## 5.0 Key Personnel

**Ange-Thierry Ishimwe** – Ishimwe is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Algorithms, Software Engineering, and Database Management courses. Ishimwe worked as a research assistant for the Embedded Systems Laboratory, where he developed a prototype for a mobile application that encrypts cryptocurrency and uses Bluetooth Low Energy for a wire transfer of cryptocurrency. He will help with the backend and the database.

**David Rowland** – Rowland is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has taken Software Engineering, Algorithms, Mobile Programming, and Database Management. During his internship with Cognizant he developed a web service application using Spring Boot, MongoDB, and Angular. He was responsible for building the database, API, and implemented the car-queue feature, classroom feature, and overall navigation of the mobile application.

**Gabriel Del Carmen** – Del Carmen is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Software Engineering, Mobile Programming and Database Management courses. He has worked as a research assistant in the Computer Science and Computer Engineering Department. He was responsible for the client side. He worked on implementing the bus line feature as well as the general user interface of the app.

**Khern Toussaint** – Toussaint is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Software Engineering, Algorithms and Programming Paradigms. Toussaint worked as a research assistant for Dr. Thompson on cryptography and also worked on personal projects involving databases and NodeJs. He will assist with the database and API development.

**Tory Adderley -** Tory is a senior studying Computer Science in the Computer Science and Computer Engineering Department at the University of Arkansas. He is a student with a passion for building applications and solving problems through use of technology. Tory has completed classes such as Artificial Intelligence, Software Engineering and Operating Systems. Along with a strong understanding of the fundamentals of computer science, Tory has built several web and mobile applications throughout his time as a student at the university. His internship experience with Engine eCommerce and with the university has also helped him to develop as a software engineer. He will be primarily responsible for building the client side applications on the web and mobile platforms.

**Tomas Stevens** – Stevens is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Database Management, Algorithms, along with Software Engineering to help benefit his team. He has worked as a student worker for the university dealing with large amounts of data within databases, websites and website protection, and security relating to such topics. Tasks he is responsible for are going to be the database management, and supporting role in the backend and security of the software, on both mobile and any other applications.

## 6.0 Facilities and Equipment

No Facilities or Equipment provided by the university were used for the creation of our application

## 7.0 References

"Spring Projects." Edited by Pivotal Software, *Spring*, 2019, spring.io/projects/spring-boot.

"The Most Popular Database for Modern Apps." Edited by MongoDB, *MongoDB*, 2019, mongodb.com/.

"Develop a Quicker, Safer End-of-Day Process for Student Pick Ups." Edited by Fetch Kids, *FetchKids School Pickup App – California*, 2019, fetchkids.com/.

"The Number One School Safety Platform." Edited by Pikmykid, *PikMyKid*, 2019, www.pikmykid.com/.

"React Native · A Framework for Building Native Apps Using React." Edited by Facebook Inc., *React Native Blog ATOM*, 2019, facebook.github.io/react-native/.

"React – A JavaScript Library for Building User Interfaces." Edited by Reactjs , *– A JavaScript Library for Building User Interfaces*, 2019, reactjs.org/.

"The Ultimate Carline Management System." Edited by Horizon Marketing, *School Dismissal Manager*, 2019, www.schooldismissalmanager.com/.

"What Is Full Stack?" Edited by W3Schools , *What Is Full Stack*, 2019, www.w3schools.com/whatis/whatis_fullstack.asp.