ABSTRACT

First responders from the Little Wabash Fire Protection District, located in Southern Illinois, often face communication problems when handling emergencies. They usually cannot coordinate their actions effectively and maintain accountability as first responders because communication only flows downward, from the chief/dispatcher to the first responders. It is possible to decrease the extent and frequency of this problem by using a mobile application that improves interdepartmental upward communication, reducing loss of life and property resulting from these incidents.
ACKNOWLEDGEMENTS

Completing this project would have been impossible without the guidance and supervision of Dr. Donald Roberts, my mentor, project advisor, and sponsor. Dr. Roberts, who is an Associate Professor of Computer Science at the University of Evansville, met with me almost every two weeks and answered enthusiastically all my questions and concerns. He also coached me on how to design and implement the application based on his personal experience in developing web and mobile applications. It was through Dr. Roberts that I came across this project since he is part of the department of first responders that it is in need of a solution to improve interdepartmental upward communication.

I am also thankful for the support of my academic advisor, Dr. Deborah Hwang, who is an Associate Professor and the Program Director of Computer Science at the University of Evansville. Despite her demanding schedule, Dr. Hwang carefully reviewed my written work and provided me with detail feedback on how to improve my methodology and make my work clear for any audience.
LIST OF FIGURES

Figure 1: System Overview Diagram
Figure 2: Service Struct Interface
Figure 3: User Interface Layout
Figure 4: Login Screen
Figure 5: Responses Screen
Figure 6: View Runs Screen
Figure 7: App’s Login Screen
Figure 8: App’s New Response Screen
Figure 9: App’s View Runs Screen
Figure 10: App’s Run Details Screen
Figure 11: App’s New Run Screen
Figure 12: App’s View Users Screen
Figure 13: App’s New User Screen
Figure 14: App’s Settings Screen

LIST OF TABLES

Table 1: Run Database Table
Table 2: User Database Table
Table 3: Response Database Table
INTRODUCTION

First responders from the Little Wabash Fire Protection District, situated in Southern Illinois, usually have interdepartmental upward communication problems when emergencies occur. They often cannot coordinate their actions effectively. Additionally, they cannot maintain accountability as first responders during these events; that is, they cannot keep track of who is responding to the emergency call until they meet in person at the scene of the emergency or at the station.

By creating an iOS mobile application that first responders can use during emergencies, the magnitude and frequency of this problem can be decreased. The mobile application will improve interdepartmental upward communication, allowing first responders to communicate with each other rapidly and with little effort. Therefore, loss of life and property resulting from these incidents will decrease.
PROBLEM STATEMENT AND BACKGROUND

According to the National First Responders Organization, a first responder is “an individual who is likely to run towards an event rather than away” [1]. A first responder is a volunteer who probably is among the first people to arrive at a scene of an emergency such as a road accident, a fire, a natural disaster, or other hazard. First responders are trained to deal with a wide variety of potential medical emergencies. Their main goal is to minimize loss of life and property resulting from these incidents.

There are thousands of local first responder departments across the United States. Over 70% of these departments are staffed by volunteers [2]. The Little Wabash Fire Protection District in Grayville, Illinois, is an example. Grayville is a small town with around 2000 inhabitants, and since it is so small, it does not have a professional fire department. Instead, Grayville relies on first responders to handle local emergencies.

Each first responder department in the United States has the autonomy to make decisions about their local communication system. The US House of Representatives estimates that “70% of public safety communications equipment is still analog, outdated, and incompatible with communications systems in neighboring local jurisdictions” [3]. The lack of proper communication systems can result in ineffective coordination, often ending in tragedies.

The Little Wabash Fire Protection District frequently encounters two-way communication problems. Their communication system focuses on transmitting information downward. In the event of an emergency, the dispatcher notifies first responders about the incident, providing relevant information. First responders are notified through an automated system that provides real-time messaging to pagers and cellphones.
On the other hand, upward communication is practically non-existent. When a first responder is notified that an emergency has occurred, he can choose to respond or ignore the emergency call, commonly known as a run. Usually, this decision is made based on where the first responder is or what he is doing. For instance, he might be out of town or at work and cannot leave. If the first responder is available, he goes to the fire station or directly to the scene of the emergency. However, neither the chief nor other responders know when someone is responding to the run and where he is going next until they see him in person. Similarly, if a first responder is not available, others do not have a way to know that he is not responding to the run.

Moreover, the ability to respond effectively and mitigate incidents requires the coordinated actions of first responders. However, when first responders cannot communicate effectively and maintain accountability, it can literally cost lives. Therefore, this communication problem not only affects the Little Wabash Fire Protection District but also affects the community. For example, if there is a huge fire at a store and only two first responders go to the scene, not only those inside the store are at risk but also those two responders because there are not enough personnel. Conversely, if there is a car accident and eighteen first responders show up at the incident, then that means the department is not using its resources well.

In the past, there has been one attempt to solve this interdepartmental communication problem for the Little Wabash Fire Protection District. In the fall of 2015, a group of students at the University of Evansville taking a software engineering class worked on a potential solution. Their project consisted in creating a responsive web application that could be used by the chief and first responders to maintain their accountability in the event of an emergency. However, the project was only web-based and never got to the level of being fully usable.
REQUIREMENTS AND SPECIFICATIONS

The sponsor required that a mobile application for both iOS and Android be developed to solve the interdepartmental communication problem that the Little Wabash Fire Protection District is currently facing. The mobile application should communicate with the server in real time, providing updated information about the emergency and first responders actions. The application also should have separate interfaces for administrators and users.

The administrator should be able to create, edit, and delete first responder’s user accounts on the mobile application. In addition, the administrator or first responder should be able to create, close, and delete runs. First responders can perform these actions in case the chief is not reachable or cannot respond to the emergency call.

When an emergency occurs, the mobile application should allow the administrator or first responder to easily create a run and immediately send a notification to everyone with an account. The application should display a list, linked to the run, of all first responders specifying who is going directly to the scene, who is going to the station, who is not responding to the run, and who has not answered yet. Additionally, a total for each category should be shown to clearly summarize how many first responders are responding to the run and what their actions are.

From the user’s perspective, a first responder should be able to get a notification of the run in real-time. When the responder opens the mobile application, it should take him directly to the current run and show him the basic information about the run. Additionally, it should allow the first responder to choose between three options: going to the scene, going to the station, or not responding. A list displaying who is responding or not to the run also should be available, so that the responder can keep track of who is going to the scene and change his decision whether he
notices more personnel is needed at a certain location or he notices enough first responders have volunteered.

In addition, the mobile applications should allow both administrators and users to log in the first time and always remain logged in unless they explicitly log out. It is important that, during an emergency, administrators and users can access the application quickly, without having to waste time logging in, so that they can create a run or respond to it in just seconds.

An optional specification is to create a web application that will also communicate with the server. The web application should have the same features as the mobile application with separate interfaces for both administrators and users. However, the priority is to create the iOS application, the Android application, and the web application, strictly in that order.

Developing this mobile application and, potentially, the web application would help coordinate the actions and maintain accountability of first responders during the event of an emergency. The use of these applications would enhance their ability to respond effectively and mitigate incidents. These applications would provide a quick and easy-to-use solution that would help improve upward communication for the Little Wabash Fire Protection District.
DESIGN APPROACH

System Overview

The frontend consists of an iOS application built using the Swift programming language. In addition, Rails was used to build the backend. The application communicates with the server over the HTTP protocol in JSON format. Both frontend and backend were built in parallel. Figure 1 provides an overview of the system architecture.

![System Overview Diagram](image)

Figure 1: System Overview Diagram
**Frontend**

The mobile application was developed in Swift 4 using the latest version of Xcode. Xcode is an integrated development environment for Mac OS used for developing software for Apple products. It also includes a simulator to build and run iOS applications.

A Model-View-Controller (MVC) approach was used to design the iOS application. The data model represents the structure of the information stored in the app. The data model is composed of two classes used to represent a run and a user and a networking singleton struct. The views were built using the storyboard. The storyboard is a file that contains the visual representation of the application’s user interface. It allows to create and show the screens of content and the transitions between them. The views were linked to view controllers. View controllers implement the app’s behavior. They coordinate the flow of information between the app’s data model and the views that display that data.

As mentioned earlier, the model consists of two classes and a struct. The Run and User classes store the information that different scenes need to display. A scene is a storyboard representation of a screen of content in the app. The Run and User classes contain different properties and one initializer. The initializer in both classes takes a JSON dictionary as a parameter, which is used to initialize each property of the class. In addition, there is a service singleton struct. The service struct encapsulates all the networking code needed to communicate with the server over the HTTP protocol using JavaScript Object Notation (JSON). The struct implements multiple methods to perform GET, PUT, POST and DELETE HTTP requests using the `URLSession` API. The `URLSession` class natively supports the data, file, ftp, http, and https URL schemes. Each method in the service singleton makes a `URLRequest` using a shared
**URLSession** data task. When the task is completed, a completion block detects if errors occurred and parses the data using the *JSONSerialization* class. This class converts JSON to a Swift dictionary. Additionally, each method takes a function as a parameter. This function is called after detecting an error or successfully parsing the data along with the results. Below, in Figure 2, is the interface for the Service singleton.

```swift
import Foundation

internal struct Service {
    internal static let sharedInstance: RespondersUI.Service
    internal let sharedSession: URLSession

    internal func fetchRuns(completion: @escaping ([Run]?, String?) -> ())
    internal func fetchUsers(completion: @escaping ([User]?, String?) -> ())
    internal func delete(path: String, completion: @escaping (String?) -> ())
    internal func post(path: String, params: [String: Any], completion: @escaping (String?) -> ())
    internal func put(path: String, params: [String: Any], completion: @escaping (String?) -> ())
}
```

Figure 2: Service Struct Interface

One of the main goals of the project is to design a mobile application with an easy-to-use user interface to provide the best user experience. Multiple simple scenes were created to allow administrators and users to perform actions such adding, editing, viewing, and deleting runs and users with little effort. These scenes are interconnected by segues, which are transitions from one scene to another in a storyboard. Figure 3 shows the layout of the user interface.
When the mobile application is first started, the user is taken to the first scene, which is the Login screen. The Login screen, shown in Figure 4, has a username and password field. Once the user has logged in for the first time, the Login screen is not displayed when the app is started unless the user had previously logged out in the Settings screen.

If there is an active run, then the next scene is the New Response screen. This screen shows basic information about the run and has the three buttons shown in Figure 5 so that the user can quickly select his response to the emergency call. This screen is also accessible from the View Runs screen in case the user wishes to change his response.
If there is not an active run, the next scene after the Login screen is the View Runs screen (Figure 6). This screen displays a list of all runs in reverse chronological order so that the most recent run is on top. If a certain run is selected, basic information about the run and a table showing a summary of the responder’s answers is displayed in the Run Details screen. The View Runs screen also has an add button and an edit button, so that the admin or first responder can create runs or delete runs from the list. If the user selects the add button, he is redirected to the Create Run screen. The Create Run screen has a grid of buttons with different run types and a description field. The run’s timestamp, active, and created by fields are automatically set when the run is created. An Edit Run screen is also shown if the user selects the Edit button in the Run Details screen.

The application has a tab bar menu at the bottom, which facilitates navigation between the View Runs, View Users, and Settings screens. If the ‘Users’ option is selected, the next screen is the View Users screen. This screen displays a list of all users ordered alphabetically and has buttons to easily add, edit and delete users. The add and edit buttons are shown only if the user is an administrator because first responders only should be able to view users.

If the add button is pressed in the View Users Screen, then the next screen is the Create User screen which contains a first, last, user ID, user type, username, and password field. There is also an Edit User screen available if the administrator selects the edit button when he is in the User Details screen.

Finally, if the ‘Settings’ option is selected from the tab bar menu, the user is taken to the
Settings screen. The Setting screen shows the user’s profile information and has a Sign Out button and a Change Password button, which allows the user to change his password.

Fields and buttons in each View are connected to their respective View Controllers through outlets and actions. An outlet is a reference to an object in a storyboard from a View Controller while an action is a piece of code that is linked to an event that can occur in the application. Several subclasses from the UIViewController and UITableViewController classes were created. Each view controller loads the data, prepares the data to be send to another view controller, calls methods from the Service Singleton, and loads Alert views when networking errors occur.

**Backend**

A Rails API-only application, which has a more limited set of middleware than normal Rails applications, was used to build the backend. Instead of using Rails to generate HTML that communicates with the server through forms and links, the Rails API-only application serves JSON resources to the iOS app’s API. Among the benefits of using a Rails API-only application were parameter parsing and authentication.

There are basically three data objects used in this project, users, runs, and responses. Models and controllers were generated for each of them. SQLite3, the default database used by Ruby on Rails, was used to store the information. The database tables for each data object are shown below in Tables 1-3.
### Run

<table>
<thead>
<tr>
<th>Field</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id (Primary Key)</td>
<td>integer</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
</tr>
<tr>
<td>runType</td>
<td>string</td>
</tr>
<tr>
<td>active</td>
<td>boolean</td>
</tr>
<tr>
<td>created_at</td>
<td>datetime</td>
</tr>
<tr>
<td>updated_at</td>
<td>datetime</td>
</tr>
</tbody>
</table>

Table 1: Run Database Table

### User

<table>
<thead>
<tr>
<th>Field</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id (Primary Key)</td>
<td>integer</td>
</tr>
<tr>
<td>first</td>
<td>string</td>
</tr>
<tr>
<td>last</td>
<td>string</td>
</tr>
<tr>
<td>userID</td>
<td>integer</td>
</tr>
<tr>
<td>phone</td>
<td>string</td>
</tr>
<tr>
<td>admin</td>
<td>boolean</td>
</tr>
<tr>
<td>username</td>
<td>string</td>
</tr>
<tr>
<td>password_digest</td>
<td>string</td>
</tr>
<tr>
<td>created_at</td>
<td>datetime</td>
</tr>
<tr>
<td>updated_at</td>
<td>datetime</td>
</tr>
</tbody>
</table>

Table 2: User Database Table

### Response

<table>
<thead>
<tr>
<th>Field</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id (Primary Key)</td>
<td>integer</td>
</tr>
<tr>
<td>response</td>
<td>string</td>
</tr>
<tr>
<td>run_id</td>
<td>integer</td>
</tr>
<tr>
<td>user_id</td>
<td>integer</td>
</tr>
<tr>
<td>created_at</td>
<td>datetime</td>
</tr>
<tr>
<td>updated_at</td>
<td>datetime</td>
</tr>
</tbody>
</table>

Table 3: Response Database Table
RESULTS

When the user first starts the iOS application, the Login Screen in Figure 7 is shown. If the user has already logged in, this screen is not displayed. The Login Screen validates the presence of a username and a password. If the authentication fails or a field is empty, an error message is displayed to inform the user.

If a run is active, the New Response screen in Figure 8 is shown immediately after the application is opened. It shows the run type, when it was created, and three buttons: Station, Scene, and Not Responding. The user can select one of these three options and his response is updated immediately.

If no run is active, the View Runs screen in Figure 9 is shown after the app is opened. The runs are listed in reversed chronological order. Each row displays the run’s type, created by, and description information. It also shows four boxes corresponding to the number of responders in each category: green (going to the station), yellow (going to the scene), red (not responding), and gray (pending). In addition, a symbol shows whether or not a run is active. This screen refreshes every 5 seconds.
If a run is selected in the View Runs screen, the Run Details screen in Figure 10 appears. This screen shows the run’s type, who created the run, when it was created, the run’s status and description. It also shows the summary with the responses in each category. Additionally, there is a Deactivate button to close the run. If the ‘Edit’ button is pressed, the user is redirected to the Edit Run screen.

![App’s Run Details Screen](Figure 10)

If the user selects the ‘+’ button in the View Runs screen, the user is taken to the New Run Screen (Figure 11). This screen has a grid of buttons with the run types: accident, assist, fire, landing zone, mutual aid, and other. It also has a text area for writing the description of the run. When the ‘Save’ button is pressed, the user is redirected to the View Runs screen.

![App’s New Run Screen](Figure 11)

The View Users Screen (Figure 12) is accessible from the tab bar menu at the bottom of the app. This screen shows the users ordered alphabetically by first name. If the person is an admin, it also allows the addition and deletion of users. If a row is selected, the user is taken to the View Details screen. This screen shows the values of the first name, last name, user ID, username, phone, and admin fields. The Edit Users Screen can be accessed by the admin from the View Details Screen.

![App’s View Users Screen](Figure 12)
If the administrator selects the ‘+’ button in the View Users screen, he is redirected to the New User Screen shown in Figure 13. This screen has text fields for filling in the first name, last name, user ID, username, password, phone number, and admin privileges of the new user. If the user is successfully saved, the person is redirected to the View Users Screen. The presence of all fields is validated except for the user ID and phone number.

Finally, the Settings screen shown in Figure 14 is accessible from the tab bar menu at the bottom of the screen. It shows information about the profile of the user such as his full name, user ID, and username. It also allows the user to change his password and to logout.
CONCLUSION

By using the iOS application, first responders with iPhones or iPads can communicate in real time. The mobile application allows them to coordinate their actions and keep track of which users are responding to an emergency call. The application has an easy-to-use interface to provide the best user experience.

Future work includes completing the user authentication process using tokens. These tokens are sent to the server on each request. Additionally, the user’s credentials (username and password) should be stored in the keychain. The keychain is an encrypted container that holds sensitive information for applications. Better network recovery actions should also be added since the app currently just displays the errors.

Since there was not enough time to develop the mobile application for both iOS and Android platforms, developing the Android version is one of the top priorities. By targeting these platforms, everybody in the department will be able to use the application. Therefore, first responders from the Little Wabash Fire Protection District will no longer experience upward communication problems.
REFERENCES


BIOGRAPHY

Daniela Fuentes is a cheerful young woman from the beautiful city of Caracas, Venezuela. She is passionate about helping others who are in need, dancing to Latin music, and traveling all over the world. After graduating from the University of Evansville, she plans to travel with her family around the United States before starting, in January, her first full-time job as a software engineer at Ciholas in Newburgh, Indiana. Daniela dreams of returning to her home country after it stabilizes and creating her own software development company. She also wants to create a program to empower young female students to pursue engineering degrees in Venezuela. After working in the software industry, she would like to become a mathematics college professor at one of the best universities in her country.

As a young girl, Daniela never really knew what she wanted to do when she grew up, but she was certain that she would pursue a career in engineering or science. She grew up with a father working in business administration and marketing and a mother in industrial relations and business coaching, so both of her parents were really surprised when she decided to follow a different career path. However, it was not until Daniela arrived at the University of Evansville that she found the perfect combination, majoring both in Computer Science and Applied Mathematics.