Windows Phone 7 AED Locator

Jeff Rose, Dustin Moore, Min Gao, Jay Seo

Introduction

According to the American Heart Association, 446,000 people in United States experience sudden cardiac arrest annually. In a situation of cardiac arrest, with each minute passing the chance of the victim surviving decreases exponentially. Early defibrillation by an Automatic External Defibrillator (also known as AED) can increase a victims chances of survival as much as 90%.

The application we built is an AED finder for Virginia Tech campus. Our client, VT Rescue, is responsible for all the AEDs installed throughout the campus buildings and our application will help manage AEDs and make them more accessible by everyone at Virginia Tech. Prior to this project, there were not any known mechanism or an equipment to help people on campus to locate an AED quickly in an emergency situation. Therefore, we believe that this application can help many people survive from a dangerous cardiac arrest situation.

Background

The application is built for Windows Phone and it was supported by Project Hawaii, a project from Microsoft that provides tools, services, and mobile-plus-cloud platforms that students need to create their applications. The Project Hawaii supplied us with four Windows Phones for each team member and access to a six month Windows Azure account that we used to build this application.

Windows Phone 7 is Microsoft's product to compete in the smartphone operating system market against iOS and Android. It boasts a simple, tiled user interface that is optimised for smaller touch-based devices. The development for Windows Phone 7 is tightly coupled with other Microsoft products like Visual Studio and Windows Azure.

The Windows Azure platform is a flexible cloud–computing platform that lets developers focus on solving problems and software logic rather than worrying about deploying, configuring, maintaining, updating and fixing the systems. We used Windows Azure hosted service to run a virtual machine that contains our application logic and complex calculations and SQL Azure as database of AEDs and on-campus buildings.

Features

When the phone application launches, the application presents the user with three large buttons. The application is designed to be simple to use in an

Figure 1. Main application menu

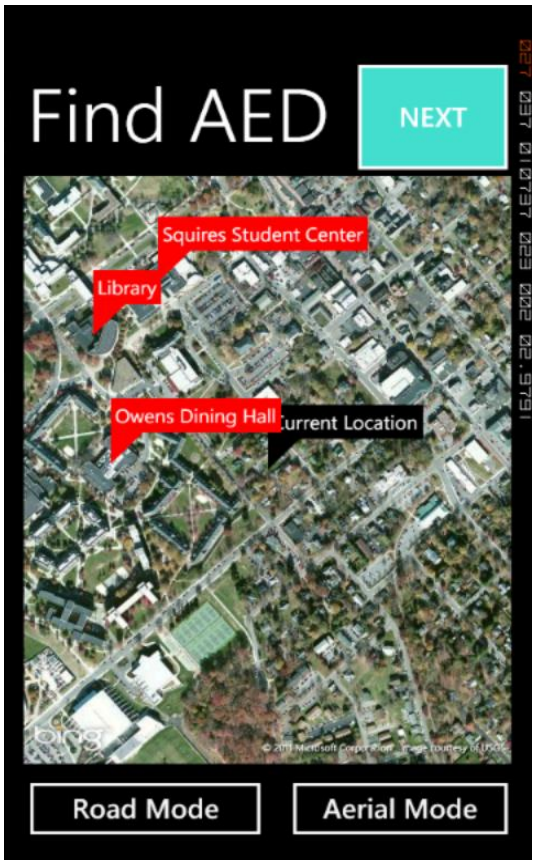

Figure 2. Building selection

emergency. Therefore users do not need prior knowledge on how to use the application and every step should be easily understood by inexperienced users. The main functionality of the application is found in the "Emergency" section. When a user selects the "Emergency" button, they are presented with simple instructions and the option to locate an AED.

When the "Find AED" option is selected, a map appears with nearby AED locations marked by orange pushpins and the user's current location marked by a black pushpin. The user then selects an AED location to navigate to by pressing the pushpin. Then, a pop-up appears with descriptions of all the AED locations for the selected building. Finally, a blue line is drawn on the map that connects the users current location to the selected building with the quickest walking directions.

The last page of the 'Emergency' section of the application asks the user to check for a pulse. Depending on what the reading was, the application instructs the user on the specific action to take.

The Settings page of the application gives the user more information about the underlying application architecture. It shows the total number of buildings on campus, the total number of AEDs on campus, and buttons to allow the user to see all of the buildings and AEDs in list format.

When displayed in the list, the buildings show their name and their coordinates for the list entry, while the AEDs show their ID number, the building that they are in, and their location description. The number of AEDs displayed on the AED Finder map is configurable through the settings page. There are two simple buttons, '+' and '-', that allow the user to increment or decrement the amount of nearby AEDs to be displayed on the map.

There is also an additional web role application that aids VT Rescue team to manage AEDs easily. It is a simple web interface that allows to add, edit, or delete an AED in the database. The application communicates to the same SQL Azure database that hosted service uses.

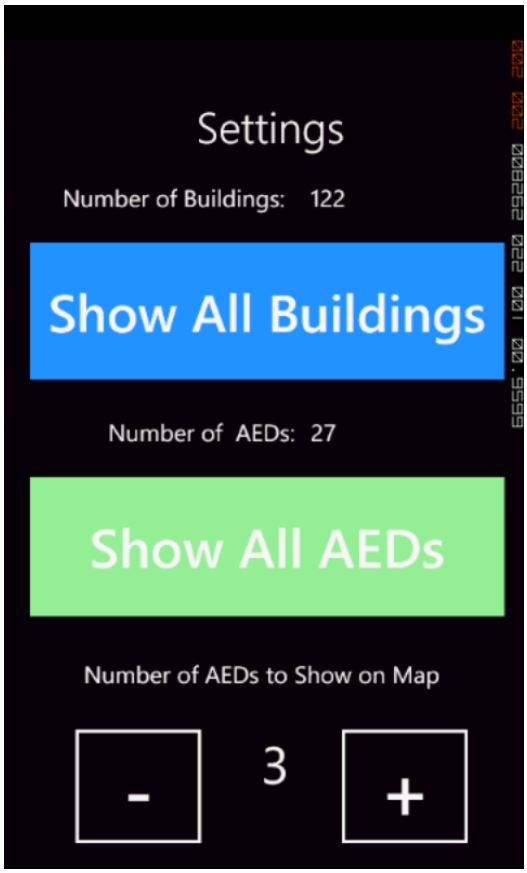

Figure 3. Settings

Technologies

Our solution was designed to be heavily based in the Microsoft Azure cloud. The use of Azure allows for the majority of computation to be offloaded to the cloud, while the phone software is focused on displaying the user interface. Windows Phone 7 communicates with the cloud service using asynchronous remote function calls, The list of AEDs and the buildings in which they reside are stored in an SQL database in Azure. Google's RESTful maps API is used to calculate walking directions.

Microsoft generously donated 4 phones to the team for our efforts in project Hawaii. The phone was the Samsung Focus, the flagship model for Windows Phone 7, which was running the Windows Phone 7.1 operating system.

Microsoft Visual Studio 2010 was used as our development environment. Visual Studio 2010 Express for Windows Phone and Microsoft Visual Web Developer 2010 Express were used for their respective platforms. Both IDEs provided great tools that created seamless integration between the phone and the cloud. For example, after the cloud service API was created, the phone application project could use Visual Studio to create a service reference that abstracted the service into a simple C# class. Features like this allowed the developer to focus on conceptual issues more often than tricky cloud server/service implementations.

Debugging this application proved to be tricky. The only way to test the web service with the phone was to deploy the application to the cloud, which was around a five minute process. Since Visual Web Developer allows the developer to launch their cloud application on their local host, a quicker solution to test the service by itself was a small C# application that called upon the local service for testing purposes.

Testing the phone application was more straightforward. We opted to deploy the application to the phone instead of the emulator for many reasons. The emulator had terrible load times and it didn't have the same location testing abilities as the device itself. Visual Studio provided debugging capabilities while the application was deployed on the device.

Implementation

The implementation of our project is primarily divided into two parts, the phone and the cloud hosted service. On the phone side, our application is structured using different buttons and views. Each button will trigger a specific popup message to inform the user what to do under which circumstance, or it will take the user to another view based on the AED use procedure provided by the VT Rescue Team. The most important portion of our application is the AEDMap view. Under this view, the cloud service was called and

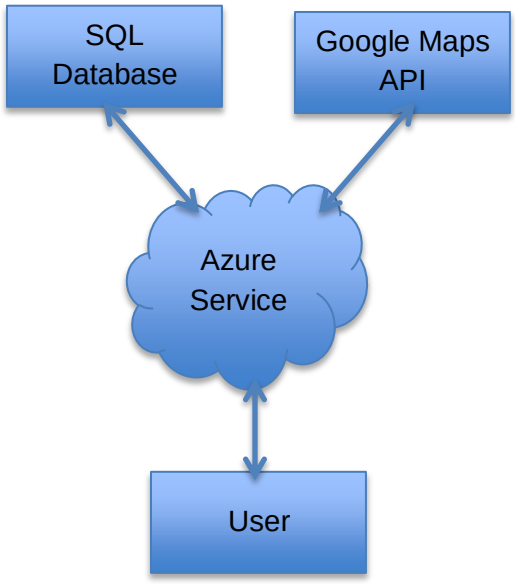

Figure 4. Application architecture

assigned local variables provided by the service.

On the cloud service side, the Azure SQL database contains two tables, one containing a list of Virginia Tech buildings and their respective coordinates and another storing AED information. The table of AEDs references the table of buildings. The selection of the closest AEDs and creation of walking directions is done by the Azure service. The service provides a function that returns the list of closest AEDs to a location and another function that returns a list of line segments representing a path between a two locations.

Function Name and Parameters	Return Values
GetAEDsForBuildings(String buildingName)	All AEDs locations within building *buildingName*
GetAllAEDs()	A list of all AEDs with details
GetAllBuildings()	A list of all buildings with details
GetClosestBuildings(Coord currentLocation, int numberOfBuildings)	Get the *numberOfBuildings* buildings closest to location *currentLocation*
GetWalkingDirections(Coord from, Coord to)	Get a walking direction from a start coordinate *from* to an end coordinate *to*

Figure 4. Azure service interface

The implementation of web interface for AED management is primarily based on ASP.NET's MVC3 framework. It enabled us to build a scalable, standards-based web application using a well-established design pattern.

Future Improvements

Bing Maps is lacking in detail compared to Google Maps when displaying Virginia Tech's campus. Bing Maps does not display walking paths or building outlines. The navigation was switched to Google Maps for more accurate walking navigation, but the graphical map display is still locked into Bing Maps on the Windows Phone platform. Improvements in Bing Maps data or a custom overlay would improve the user experience.

Currently, the application is fully reliant on the cloud service. As a result, if there's no data connection on the device, the app will not be able to function properly. A solution to this problem would be to store a local database of AEDs on the phone. This database would be updated regularly and synced with the database in the cloud. A local database would provide accurate information even when no reliable Internet connection is present.

The strong integration with the cloud service on a highly connected campus, however, opens up a world of possibilities for this application. For example, more complicated campus geographical data could be stored on the cloud and used for more advanced AED finding algorithms. One such improvement would be to factor in which floors the AEDs are located. The current application only accounts for horizontal distance. An improved algorithm would assign a higher cost to AEDs many floors above ground level.

In a life-or-death situation, the quality of the directions to the nearest AED become important. A future improvement to the walking directions would be to provide

directions while inside the building that the user selected. For example, when the user arrives at the building that they selected, the next instruction given would be, “Walk down the staircase to the first floor.” A feature like this would add utility to the application because it would provide a service that isn’t available through other applications. It is also the next step to provide the user an end-to-end emergency service.

Conclusion

The VT Rescue Windows Phone application project was academically and practically satisfying. The opportunity to use the Windows Phone, Windows Azure, and the link between them was a valuable experience for the team as computer science students. The immense processing power in the cloud allows applications on smaller, Internet-connected devices to have greater functionality. Futhermore, in the next generation of computing, the true power of mobile devices can only be unlocked if back-end application processing is used.