Project Hermes

Formerly known as CallForCode

Design Document

May 2019 - Project 40

Advisor

Diane Rover

Team

David Boschwitz - Team Lead

Caleb Nash - Lead Frontend

Logan Fladung - Subject Matter Expert & Graphics

Justin Kaufer - QA & Test Lead

Austin Keen - Design Lead

Robert Schedler - Backend & Networking Lead

Contact

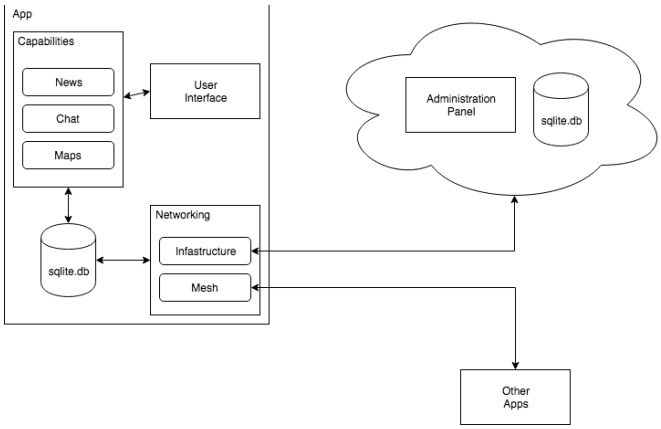
callforcode@iastate.edu

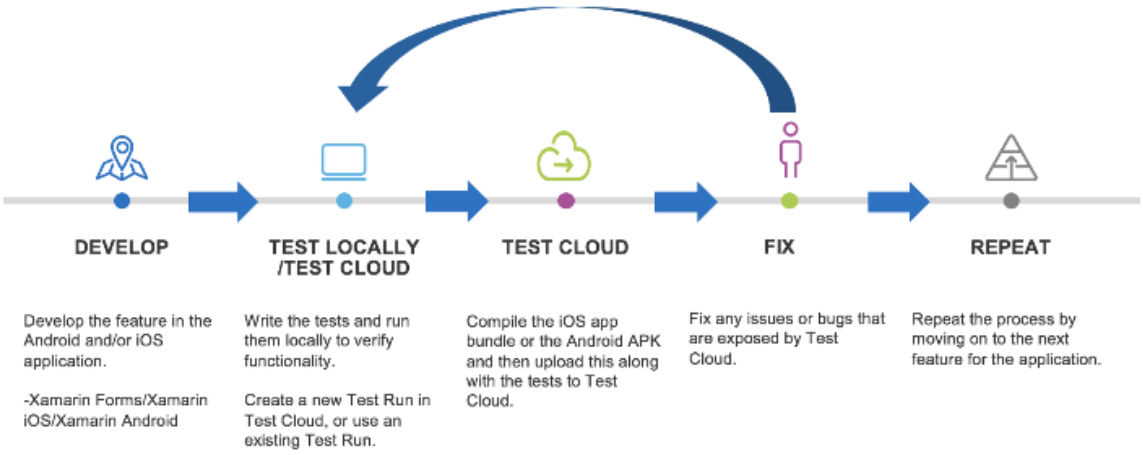
sdmay19-40.sd.ece.iastate.edu

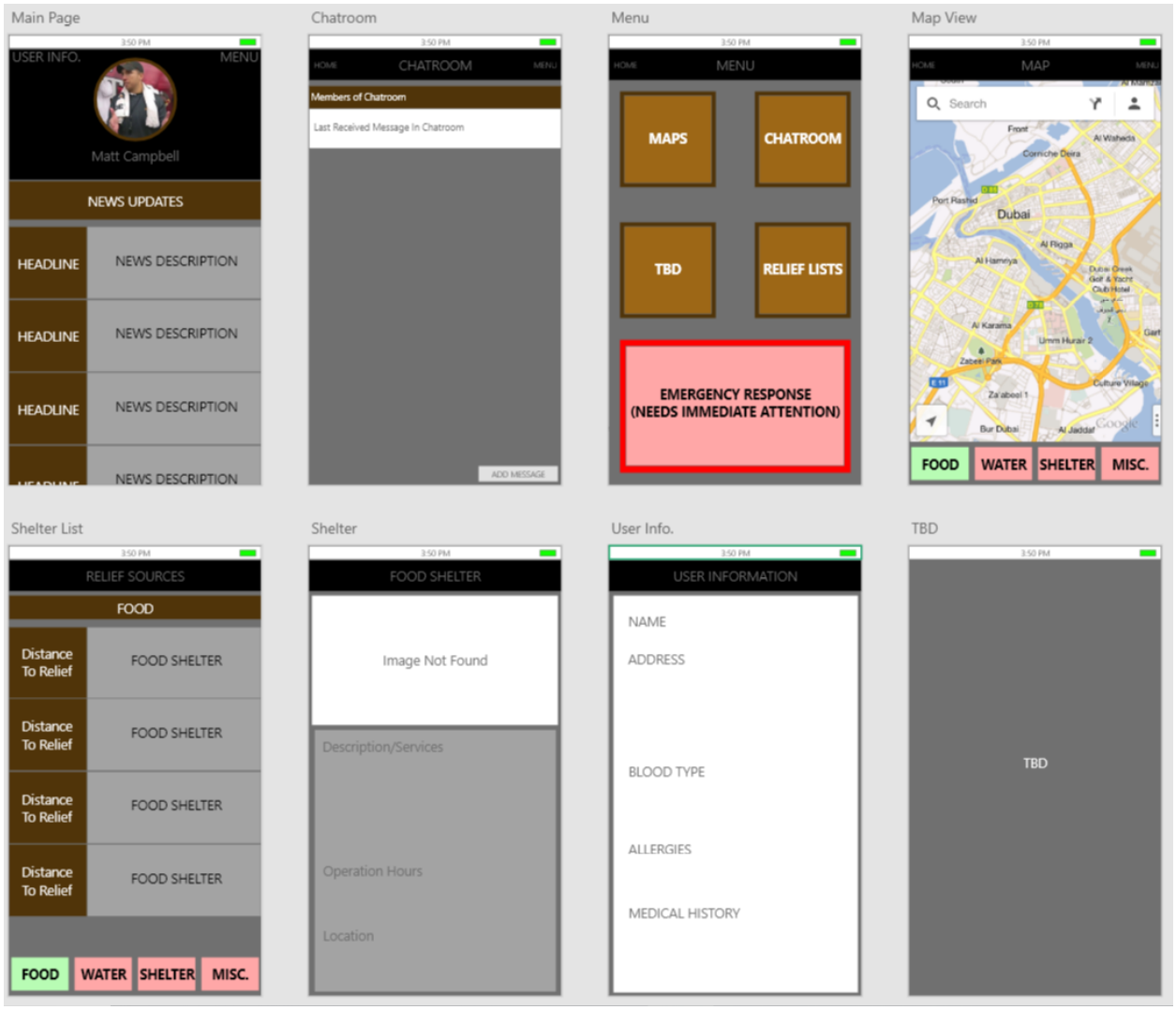
Revised: 2018-10-12/V1

Table of Contents

List of Figures







List of Tables

List of Definitions

.NET: The Microsoft utility library used commonly with C#

Xamarin: A C#, cross-platform mobile development toolkit

XAML: C# UI Templates based on/similar to XML/HTML

ECE/ECPE: ISU Department of Electrical, Computer, and Software Engineering

MVVM: Model-View-ViewModel, a frontend design pattern commonly implemented when using XAML and C#

1 Introduction

1.1 Acknowledgement

We would like to convey our gratitude towards Dr. Diane Rover, our advisor that will be guiding us through this entire project. Her project management expertise has been monumental in helping us get on track and stay on track. We would also like to thank Dr. Nicholas Fila, who is assisting us on developing user requirements. His knowledge of the subject has not only helped us tailor our requirements, but Nic is also helping us understand his thought process so we can apply this knowledge in the future.

This project was inspired by IBM's Call for Code challenge, which is "a rallying cry to developers to use their skills and mastery of the latest technologies to drive positive and long-lasting change across the world with their code." Unfortunately, the timeline of the class did not allow for our team to be involved, but this challenge helped stem the idea for this project.

1.2 Problem and Project Statement

Problem Statement

With natural disasters being as common as they are, ways to provide relief to these victims are becoming more and more necessary. The goal of Call for Code project is to design a tool to improve preparedness for natural disasters and relief when they hit in order to safeguard the health and well-being of communities. There are millions of people affected every year and without the proper tools and education, the recovery effort is slow.

Problem Solution

Our solution to this project is simple. Our first step will be to conduct user research and compile user requirements. This will be done over the course of a few weeks with the help from one of our advisors, Nicholas. After compiling user research and requirements, we will spend some time creating software requirements from the user stories. This will help us streamline what features we want to get done first and when we get into the development process we will be able to complete and test features faster. After developing and testing code, more vigorous code reviews and user tests will help us wrap up and finalize the software.

1.3 Operational Environment

The operating environment for this product will all be either server-side or on a physical device such as a cell phone. We plan on supporting multiple platforms such as Windows, Android, and iOS using Xamarin. Since we are writing software using cross-platform support, the environment

doesn't play a major part in what we are doing. We will be taking advantage of the EcE servers, which are not only out of our reach, but are very well maintained.

1.4 Intended Users and Uses

We have a wide variety of intended users, however the key commonality is that they're victims of natural disasters. We will have users of different ages, cultures, and mental statuses. We do not know how the disaster will affect each of our users, and we will focus on helping them out as best as possible. Our main focus, if we had to choose, would most likely be families with children, therefore, we are focusing most on the general natural disaster relief plans that aid families, such as shelter, food, clean water, and medical assistance. Knowing how dangerous hazardous conditions are to children and the elderly will also be a very high priority. We need the elderly to be able to easily handle using our application so that they can get to safety as soon as possible.

Our intended use is pretty self-explanatory, we are creating an application to aid those being affected by natural disasters. This application will be used as a guide for finding the relief stations that will help save our users' lives. Our application will allow users to safely route their efforts to get to the nearest shelters for food, medical assistance, and a place to rest with a roof over their heads.

1.5 Assumptions and Limitations

Assumptions:

- The EcE system will be available 24/7 and will not have any technical problems regarding loss of data or downtime.
- The software will be used in a location that has Google maps data
- The software will be used in a location that has access to NOAA/other government organization disaster evacuation maps.
- The user knows how to use a smartphone and doesn't need assistance in using any of the features.
- The user is proficient in english enough to utilize the software.
- People utilizing specific features (e.g. shelter hosting / energy hosting) are not malicious in their intent during a natural disaster.

Limitations:

- Server limitations. If the EcE system isn't sufficient, then we will have to find a different server solution and possibly receive resources from IBM.
- Network connectivity, especially in times of disaster, will not always be available.
- The system will be a mobile application used on mobile devices that can properly run a mobile application distributed through the play store/iTunes.

1.6 Expected End Product and Deliverables

CallForCode Mobile Application

The client will be delivered a mobile app with that fulfills the specified user need statements and functional requirements specified in the team's initial research. It will provide users access to maps showing closed roads, hazardous areas, and information pertaining to relief centers and supplies. This will be delivered two weeks before the end of Spring semester 2019. (April 26th, 2019)

CallForCode Administrator Tool

Another tool that we'll provide is an application for administrators to update information seen by the users. This tool will have direct access to the database and the users will be responsible for updating data relevant to different areas affected by disasters. It will also be delivered two weeks before the Spring semester ends. (April 26th, 2019)

CallForCode Documentation

We will deliver the detailed documentation. The application will be delivered with documentation for a new software team to continue development of the project, as well as documentation for an administrator to run the app, and proper documentation for a user to use the app. It will detail how administrators will be able to update databases and troubleshoot issues. It will also have details about how the main application is laid out and how users can access all the information. To be delivered two weeks before the end of the Spring semester. (April 26th, 2019)

This product will not be commercialized.

2. Specifications and Analysis

2.1 Non-Functional Requirements

NFR1. Security

- Secure sign-in of authenticated users cannot be duped
- 100% of messages will be encrypted

NFR2. Performance

- It is important that users experience short response time, within 3 seconds from request
- High availability of our system with above 99% uptime

NFR3. Accessibility

- Ease of usability, the interface is easy to learn and navigate
- Application should work with or without network connectivity

NFR4. Scalability

- The ability to maintain performance and operation from not only concentrated local areas but to more distributed geographic regions
- Ability to add new features and deploy rapidly

2.2 Functional Requirements

FR1. Provide users with locations of relief support, such as medical assistance, food, shelter, etc.

- This focuses on providing help to those that have means of transportation to allow them access to crucial items needed for survival.

FR2. Mark roads that are unsafe and not allow routes through them

- Hazardous roads are another safety measure put in place to ensure our users are not put in a position where they are in danger or have to reroute themselves

FR3. Track users and reroute paths based on traffic along popular routes

- This helps the users efficiently get to their destination which is beneficial for time sensitive situations.

FR4. Allow users to communicate with each other via chat messaging within the application

FR5. Function even when internet and cellular connections are unavailable through an ad hoc network process

- The ad hoc network allows information to ping off of other users to transmit the most up to date data throughout the application

FR6. Work across iOS and Android platforms

- The application shall be designed to function cross platform

FR7. Have user status' for different kinds of users, including relief worker, ordinary user, and admin

- This will allow the functionality and purpose of the app to differentiate based on the user's intent

FR8. Allow users to send emergency pings to relief workers

- A call for immediate assistance in the event that there are relief parties nearby or search teams out

FR9. Provide a constant news update based off locational settings

2.3 Proposed Design

The current design is based on several ideals based on user research. Cross-Compatibility, Lightweight on System Storage, use in areas with little-to-no access to internet, capabilities that solve common problems for people.

Cross-Compatibility is solved by using Xamarin. Xamarin is a library for C#/.NET that allows for easy cross platform development. It has proved its worth over several years. During some initial research we found that it has been used for many apps with similar functionality, including a prototype with mesh networking capabilities.

Lightweight on system storage is solved by using sqlite, a database system that is extremely light weight.

Use in areas with little-to-no-access of internet is solved using our plan for a networking component that will leverage normal internet as well as internet over ad-hoc networking. This is still in the research and viability phase.

Capabilities to common problems were create using user research to discover if we were correctly anticipating our users needs. After analysis of common large scale disasters we looked at the response that was needed by humanitarian relief efforts and what else could be used. We determined that we should create three components initially: news, chat, and maps. The news app will include a way to easily send important, validated updates over the air to the masses quickly. The chat app will include a way to send messages and SOS pings over the air using all network capabilities asynchronously. The Maps app will include offline maps with updates to roads and routes of exit over the air.

2.4 Design Analysis

Figure 2.4.1

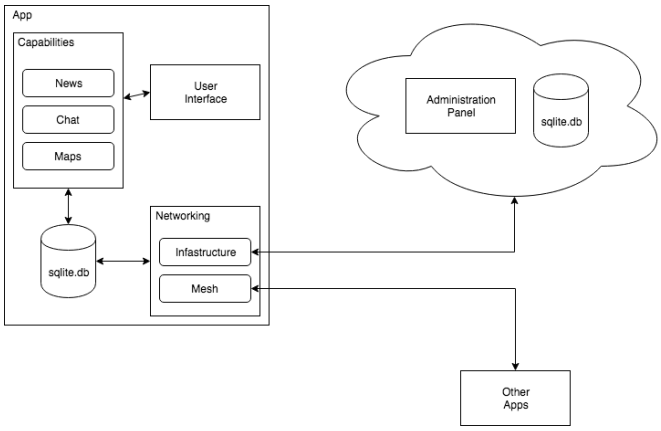


Figure 2.4.1 describes the application architecture in a large grain. We have condensed what we believe user needs are into the “Capabilities” box in the diagram. All other features of the drawing support the capabilities listed in the “Capabilities” box . We looked into how natural disasters are handled from a humanitarian point of view and tried to create a solution to some of the problems we believe that existed. We also looked into the plausibility of a mesh network using Wifi Direct, BlueTooth, and/or other communication interfaces using Xamarin.

We think a peer-to-peer network could be very powerful in helping communities communicate, and the Red-Cross agreed with us. It is not uncommon for network infrastructure to be disabled after a natural disaster so allowing communities to still have some form of communication is much better than none. The peer-to-peer (or ad-hoc) network brings with it a discussion about security that we have yet to solve. For example how can we prevent Alice from looking at Bob and Jane’s conversation although due to the nature of peer-to-peer Alice has the possibility of transporting the latter messages between Bob and Jane. currently we lack a solution to authenticate users. Another weakness we may have is that our research on natural disasters and our solutions may not work well when applied in a real natural disaster situation, although we have conducted extensive reader to mitigate this risk

3 Testing and Implementation

3.1 Interface Specifications

Software-Software

Since we don't have any hardware, we will be using exclusively software-software interfacing testing. We will be writing all of our tests in-house to test connections between our devices and make sure that our devices are communicating and passing data correctly. Below are the different tests that we will have to implement throughout the course of our development process.

- Application to Cloud (client to server)
- Cloud to Application (server to client)
- Mesh Network - Application to Application (client to client)

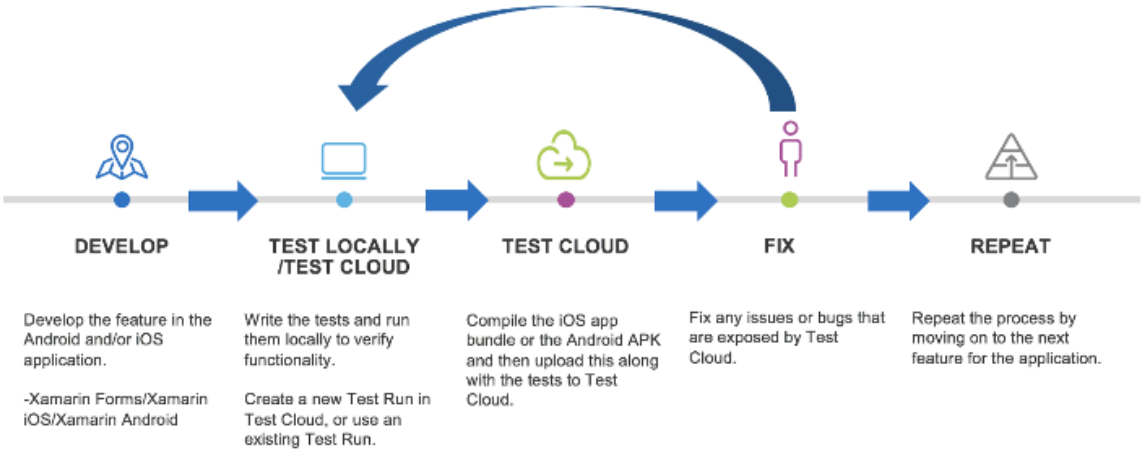
3.2 Hardware and software

Xamarin
Xamarin uses a C#-shared codebase, which allows us to use Xamarin tools to write native Android, iOS, and Windows apps with native user interfaces and share code across multiple platforms. We will be using this to make it easier to write a solution that will be available on both iOS and Android.

We will most likely not need any additional software to run tests in Xamarin. We will be able to write tests locally to verify their functionality, and be able to implement them either as standalone tests or test them on something like a test cloud that will test the code on a number of devices at the same time. Eventually, as I said we might implement some software to help us run these tests on a number of virtual devices. A service or software like the one mentioned would be in the future and is not necessarily needed as we have a number of devices to test on, as I touch on below.

We will be our own phones that each one of team members have as hardware when we are testing our solution. We have a wide range of devices that will help to deal with version control. This way, we can make sure that whether a client's phone is Android or iOS, or various different versions of these operating systems, that we will able to provide those people with the full functionality of our application.

Figure 3.2.1



3.3 Functional Testing

Unit Testing

Unit testing allows developers to test their code down to the smallest units possible to assure they are working as expected before trying to implement several pieces together. As a team there will be a standard format for unit testing of arrange, act, and assert to keep tests consistent and easily observed by others. Directed by our test engineer, the unit tests will be expected to be written by each developer as they are writing code, not at the end. This will ensure the correctness and functionality to catch any mistakes before they create a domino effect. All components of functional requirements shall be extensively unit tested to ensure quality. Success of the individual test is determined by the actual output matching the expected output. Overall success of testing a feature is measured by the percent of tests passing.

E.g. If someone creates a new user, does that user get saved in the database with the correct information.
Arrange: Set up mock database
Act: Create a new user with the constructor injection
Assert: NUnit assertion comparing the data of the user

Integration Testing

Integration tests will be made throughout the development process, along with the unit tests. The difference between these and the unit tests is that these test functionality of code as a group. The purpose of integration testing is to test faults on the interaction between specific functions. The team will use the Xamarin.UITest framework combined with NUnit to perform these tests on interfaces. The format will follow the same setup as unit tests in the order of arrange, act, and assert to maintain uniformity. Success is determined by the test correctly executing the function that is designed by the test.

E.g. If a user wants to navigate to the news feed from the main menu, assuming they click the correct buttons do they reach the desired page.
Arrange: Instantiate the app on main menu
Act: Navigate the app using Xamarin REPL, and save to test case
Assert: NUnit assertion confirming pageID

Xamarin.UITest framework has a very useful function of saving a list of terminal commands on the UI components and compiling them into code to be implemented in the test you're writing. This way we will be able to navigate a function how we'd like to test and save it to be automated in our tests.

System Testing
These will be made towards the end of the development process, as this is to make sure that our functions and applications fully work with the systems that we are running them on. An example of this would be to run specific tests to make sure that specific functions work when using Android and iOS API's on different phones. That way we know that it doesn't matter what phone the user is using, they will be able to use the application to its fullest.

Acceptance testing
This is a type of testing that our team most likely will not be using, due to the fact that we will only be installing the application on mobile devices, or running them locally on an emulator.

3.4 Non-Functional Testing

Usability Testing
These tests will not be written in code, but will be research on the user experience of the application. Usability by all types of users is extremely crucial to the application. A variety of procedures established by the team or developer responsible for the feature will be established and tested by various populations. The test engineer will establish a survey to ensure that a variety of demographic and skill sets are being tested. Any feedback received from usability tests will be accounted for and used to help redesign or validate the specific feature. Success will eventually be determined by all populations being able to successfully complete the tasks established by the team.

E.g. A user should be able to create a their own user then go in and set their location settings.

A usability testing model will be used to first determine the demographic being tested. Then after requesting certain tasks within the application, the users will give specific feedback on their experience performing the function, as well as any feedback not pertaining to the specific task.

Security testing
With the application being user to user communication based, security is a big concern. Every time the application gains a feature or update the reassurance of security will need to be confirmed. The connection between the application and the cloud will also have an encryption to

ensure user privacy. Manual testing will be used to make sure all connections are sound and cannot be translated or intercepted.

Performance testing

The performance testing will revolve around accurate data being translated between users. The accuracy as well as efficiency will determine the effectiveness of the application. Since timely communication of data is the whole basis of the app, there will need to be many tests implemented to assure that. Establishing performance metrics is key to make sure there is no bottlenecking or false information.

> E.g. Testing to assure the influx of updated maps data is loaded to a user's application in a timely manner. Success is measured by a predetermined timeout established by the feature team.

3.5 Process

For project process flow diagram, see Appendix 4.3.1

Testing in our current state has been difficult. Most of it has occurred when selecting protocols for mesh-networking. We tested this by ensuring our phones could communicate basic information even without cellular or data coverage. Our UI/UX wasn't tested so much as our team came to a consensus and agreed upon our prototypes. At the moment we have nothing developed, therefore nothing tested. The following are tentative methods for future versions:

Features

The features will be tested using unit tests and the black box model when applicable. There will also be a heavy focus on the performance and usability aspects of the features, to make sure they incorporate successfully into the project plan.

Cloud/Database

This will be a mixture of unit tests for the backend framework, and security testing the communication between the individual's application.

User to User Communication

Security testing will be very important to assure privacy. We will run tests similar to penetration testing to ensure that our database and users are secure.

3.6 Results

So far, we haven't done very much developing, because this application has required us to learn several new programming languages, and complete a lot of research to understand how we will go about generating an application that will best benefit the relief groups and their relief efforts. We do have a functioning UI prototype (Figure 4.3.1), however it only allows us to maneuver

throughout the application, but not change states. We will be taking this prototype and using it as a guide as this project progresses, we have much more work to do and will continue to build this application step by step and constantly test our application both with user focus groups and programmable test cases.

As for implementation issues and challenges that our team has been facing so far during this first semester, there have not been many. This is because we are focusing on outlining our project and making sure our user requirements are solid.

That doesn't mean that we haven't ran into any roadblocks, however. We have ran into some implementation issues in regard to our plan for a mesh network. Although we are using Xamarin to help aide us in making a cross platform application between iOS and Android, we still have not successfully came up with a solution to our mesh network problem to the point where it will work smoothly across all devices. We have, however, found a solution that will work for our mesh network on all Android devices.

We have decided as a group that if we cannot feasibly find a solution to our mesh network problem with regard to all iOS devices, then we will either shelf the iOS end of the application or not include mesh network options for the iOS users. This is not optimal but will be at the discretion of what we feel will be best. We learned with this roadblock that all your plans at the beginning of the project sometimes will not iron out and will need to be changed.

4 Closing Material

4.1 Conclusion

Thus far in the project, we have researched natural disaster relief efforts, formulated user requirements, built a project plan, and gathered all necessary tools to begin constructing the application. Our next step is to communicate with experts in the natural disaster relief field, and hear their opinion regarding whether our idea for this application would be effective/beneficial. Another next step will be researching mesh networking with Xamarin, creating mock UX/UI designs, and completing other necessary research to better understand how we are going to build this application.

As for our plan on building this application, we will split up into teams based on our expertise. Some of our members focus on front-end, and others on the back-end, therefore we will be splitting up into teams of 2-3 and each team will focus on the full stack implementation of a specific functionality. Outside of the functionality teams, we will have leads that will be in charge of specific areas of the overall development. The leads will be responsible for answering any inquiries that functionality teams may have, for example, Austin will be responsible for answering any UI/UX inquiries, while Justin is responsible for answering any testing inquiries. It is important to have these leads, because having consistency is crucial to reducing/locating problems in the application.

This is the best course of action because it will allow our team members to maximize their productivity by focusing their specializations, as well as not having too many programmers working on the same problem. Also, less programmers editing code within each functionality will make it easier to find problems.

4.2 References

1. "Register for the Challenge." *Call for Code*, callforcode.org/challenge/.
2. "Rebuilding Rail after the Earthquake." *Great Journeys of New Zealand*, www.greatjourneysofnz.co.nz/blog/kaikoura-earthquake-derails-the-main-north-line/.
3. Roy, Eleanor Ainge. "New Zealand Earthquake: First Relief Trucks Sent to Kaikoura as Road Opens." *The Guardian*, Guardian News and Media, 17 Nov. 2016, www.theguardian.com/world/2016/nov/17/new-zealand-earthquake-first-relief-trucks-sent-to-kaikoura-as-road-opens.
4. "The Serval Project." *Serval*, www.servalproject.org/.
5. Pokharel, Govind Raj. "Nepal Earthquake 2015 Post Disaster Needs Assessment." *Www.nepalhousingreconstruction.org*, Government of Nepal, 2015, www.nepalhousingreconstruction.org/sites/nuh/files/2017-03/PDNA%20Volume%20A%20Final.pdf.
6. Lokesh, Likitha. "Mobile Application Testing with Microsoft's Xamarin Test Cloud Services (Part 1)." *Medium*, Slalom Engineering, 15 Mar. 2018, medium.com/slalom-engineering/mobile-app-testing-with-microsoft-test-cloud-services-part-1-cob40b7c19d0.

4.3 Appendices

4.3.1 Project Process Flowchart