SANTA CLARA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: June 2, 2021

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Sarah Ahmed
Marco Marenzi
Leila Scola

ENTITLED

MARTHA: Offline Streaming Media for Cameroonian Refugees

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREES OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING
BACHELOR OF SCIENCE IN WEB DESIGN AND ENGINEERING

Thesis Advisor

Nam Ling
Nam Ling (Jun 9, 2021 09:03 PDT)

Department Chair

MARTHA: Offline Streaming Media for Cameroonian Refugees

by

Sarah Ahmed
Marco Marenzi
Leila Scola

Submitted in partial fulfillment of the requirements
for the degrees of
Bachelor of Science in Computer Science and Engineering
Bachelor of Science in Web Design and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 2, 2021

MARTHA: Offline Streaming Media for Cameroonian Refugees

Sarah Ahmed
Marco Marenzi
Leila Scola

Department of Computer Science and Engineering
Santa Clara University
June 2, 2021

ABSTRACT

MARTHA is a mobile application intended to deliver educational content to refugees from Cameroon currently residing in Nigeria. Refugees were forced to flee after advocating for educational freedom, so education is of utmost importance to those residing in these refugee camps. Parents want their children to benefit from learning general educational content, as well as vocational content which is important for social mobility. Our application will display files that can be followed at the users' pace. While the project entails two phases, the final goal is to have a Raspberry Pi act as a server and local database, allowing local Android tablets to stream content from its own ad hoc wireless network. Our system will be able to provide equity in terms of social and economic advancement to those who need it the most.

MARTHA: Offline Streaming Media for Cameroonian Refugees

by

Sarah Ahmed
Marco Marenzi
Leila Scola

Submitted in partial fulfillment of the requirements
for the degrees of
Bachelor of Science in Computer Science and Engineering
Bachelor of Science in Web Design and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 2, 2021

Acknowledgements

We are profoundly thankful to everyone who helped us through the development of this project. A special thank you to our advisor, Dr. Figueira who has been a guiding light since the project's conception. Also, Allan Baez Morales for his wonderful insight and ingenuity. A huge thank you to Isaac Zama from the Victoria Relief Foundation for his guidance, foresight, and passion. We would also like to thank Dr. Parker and our teammates from EWH for their contributions. Finally, thank you to Santa Clara University School of Engineering for helping fund this project.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

Sadly, like many African nations, Cameroon was passed between imperialist hands several times leaving its infrastructure fragmented in many ways. Once it gained independence from both France and the United Kingdom in the mid-1900's, political power became consolidated to one ruling party and saw little turnover. Eventually the English-speaking north advocated for more freedom, as well as the transition to English being the primary language taught in schools. This unrest, coupled with the tradition of guerilla warfare to gain freedom, led to war between the majority of the west and north, which declared itself Ambazonia, and the Cameroonian military.

Ambazonia now contains roughly 679,000 people displaced from their home which has led to about 60,000 refugees seeking safety in Nigeria. About 80% of schools in Ambazonia have been shut down and supplies have been stopped from entering. Nigerian neighbors have graciously been setting up camps, with about 89 in existence, and welcoming Cameroonians into their communities. Supplies are spread thin. Necessities are provided by refugee organizations and the Nigerian government, including living accommodations, food, and water. Tarps or dwellings of mud-bricks are set up and refugees earn money for food by working as agricultural day laborers. Parents often must pay a small fee to allow their children to go to school. Given that refugee camps are continuously supplying food, water, and shelter, education seems to be a primary need of refugees that is unattainable. Cameroon and Nigeria have the highest rates of educational enrollment. Cameroon sits at 93% and Nigeria at 98%, and education is valuable to Cameroonian refugees.

Without education, parents fear their children will lack the skills to perform well in the current job market. They have fled their country to provide a better life for themselves and their families. Without an education they are stuck without skills to earn a living. Education also provides the means to understand proper hygiene, bargaining skills, and communication skills. If action is not taken to provide access to education, refugees are stuck in a negative feedback loop of lack of knowledge that leads to lack of jobs that leads to lack of money to buy an education. Without an education, long-term success is impossible to attain.

While there is a large focus on educating the next generation, this has been made extremely difficult by the lack of educational resources, as well as lack of educators in these camps. Most camps have a single educator, if they are lucky, and only a blackboard to teach with, lacking things as basic as pencils and papers. Virtual learning content can be easily distributed to many, filling the role of educators, and eliminating the need for physical resources, since monetary barriers prevent access to school supplies. In addition, there is poor Internet connection and when connected, it is extremely expensive to download content. Any current virtual learning environments are more expensive than public school education in Nigeria, and therefore, inaccessible. The lack of Internet access in these refugee camps thwarts the implementation of any solution which utilizes electronic devices that rely on Internet connectivity as a means of distributing content. There seem to be no affordable outlets for young refugees to learn how to provide for themselves and their community which can be utilized offline.

1.2 Stakeholders

Our team identified the following stakeholders in our project:

- Cameroonian refugees youth and rural community.
- Victoria Relief Foundation.
- Engineering World Health SCU.
- SCU Frugal Innovation Hub.
- Academe Nursing Institute (Vocational Education School), Ikom, Nigeria.
- Community Career Academy (Vocational and General Education), Ikom, Nigeria.

1.3 Existing Solutions

There are very few educational tools that are currently available to be utilized by children in Nigerian refugee camps. These families and children travel with bare necessities. Agencies that support the refugees camps struggle to provide food, water, and shelter. Learning mediums are often restricted to pencils, paper, and maybe a teacher and classroom with a blackboard.

Various learning tools do exist that function without the need to connect to Internet often. Obviously, written curriculum can be acquired. Besides that, virtual learning materials are often apps that can be downloaded onto mobile devices. They often function as interactive games or learning material and work best with one person using the app at a time. Other solutions include eBooks, tutorial videos, notes, quizzes, career guidance, and other less dynamic

services. These are often free to use and do not require internet access, but access to premium features and content that will truly provide the learning material requires users to pay additional fees.

None of these learning tools provides a system that can not only download educational material, but then distribute it across many devices and provide a learning platform for many students at one time. Lack of money or internet access should not hinder someone from accessing educational tools that can allow them to achieve their full potential.

1.4 Our Solution

We will provide an affordable, reusable, sustainable, and long-term solution to this issue. We propose an educational application that can be utilized offline through an ad hoc wireless network as a means of delivering educational material between electronic devices, which the young refugees will use as vehicles for learning. Our system will be implemented in two phases. The first phase will develop the front end of the application and connect it to a persistent database that administration can upload educational content to. The second phase will install and configure a Raspberry Pi to act as an intermediate local server and database that multiple Android tablets may connect to.

The first segment of our proposal is to implement an operational application for iOS and Android mobile devices by using React Native to display our interactive learning material. It will display different grade levels to choose from and then topics within each level. Content will largely consist of text material and dynamic testing and learning modules. A single device will provide general education, vocational training, hygiene knowledge, and community-building skills. In this application, one of the main tasks is going to be storing data and sending data that may be in the data format of img(image), PDF(text document), pptx(powerpoint), mp3(audio files), mpeg(video files). In this first phase, we will use a MySQL database and PHP to configure our data and store it persistently. We will use React Native language to manipulate and transfer the data. As mentioned, the data will consist of education material and media that will instruct the students in the refugee camp.

The second part of the solution is to integrate a Raspberry Pi computer that can serve as a local database in the refugee camps that periodically retrieves updated media once it has Internet connection. With an attached wireless card, the Raspberry Pi is expected to be able to transfer data and provide an ad hoc wireless network that mobile devices can connect to while offline. This will streamline our solution and make it more robust and scalable. In addition, we will integrate a new persistent database and query language to reduce the lag in content retrieval by storing data closer to the user. We will use Strapi with GraphQL and SQLite plug-ins as our back-end content management system acting primarily as a content repository. In conjunction, we will use a Digital Ocean Cloud Server to retrieve data from the Strapi database. Ultimately the Raspberry Pi will pull its content from the Digital Ocean Cloud Server when it connects to the Internet for the most recent update. As a final update to our system, the application will be configured to allow the user to manage memory usage. In addition, we will modify our system to use solar power to recharge.

These edits to our system will create a more robust solution.

Cameroonian refugees are facing an unjust amount of difficulty. Thankfully, many of their primary needs are being taken care of by the Nigerian government and refugees assistance organizations, including shelter, food, and water. These refugees work menial jobs to provide for their children and want them to get an education. The war began over the fight for an education. We want to do our part to aid them in providing them with the resources to elevate their status. Given the current constraints of monetary barriers and Internet access, we have developed a solution that will be free to use, easy to distribute, and will not need Internet access to remain relevant and useful. Our solution circumvents the two main barriers of entry to an education: money and the Internet. Overall, we will help subside the stem of educational access.

1.5 Alternative Solutions

Most existing solution for delivering educational content require limited internet access such as our solution does, but they lack the infrastructure to implement our solution. Our solution creates a server, ad hoc WiFi network, and educational content that can be used by many at once. Other educational content requires individual mobile devices to download or stream the educational content directly and can only be effectively utilized by one user at a time.

One alternative solution is RACHEL, a plug-and-play server that stores educational websites that is streamable online on a local connection. RACHEL is a tool for the 53 percent of the world without an internet connection to access the best free educational resources [1]. RACHEL models range from 170 to 500 dollars. However, the creators also have an open-source tool, OER2Go, which is a collection of educational websites repackaged for download and online use. This system is too expensive for those living in refugees camps and the open-source tool lacks the infrastructure necessary to delivery content to the refugee camps.

Chapter 2

Requirements

2.1 Functional

We found that there are four requirements that need to be met in order to ensure that the functionality of our system is full-proof. The functional requirements in order of importance are:

1. The system will deliver content to students through Android tablets.
2. The tablets will download content from an external database.
3. The Raspberry Pi will be configured to seamlessly and autonomously update its content by pulling from a larger online web database when it is able to connect to the Internet.

2.2 Nonfunctional

These requirements are aimed at improving the user's experience. They will be verified through dialog with the client throughout the system's development. The non-functional requirements in order of importance are:

1. Sufficient educational content should be available without the need for Internet connection.
2. The server will have the capacity to store all content that is uploaded to the database.
3. The system will allow multiple users to connect to the server and retrieve data at any time.
4. The media will load in a reasonable amount of time and play with limited lag. We aim for all media to load in no more than 10 seconds.
5. The system will respond to user action in a reasonable amount of time. We aim for all reactions to occur in no more than 10 seconds.
6. The system will be accessible and useful for people with limited to no technical experience.

7. The application user interface will be simple and intuitive.

8. The system will be easy to maintain and update by outside programmers and administration.

2.3 Constraints

The system constraints are:

1. In order to distribute the technology to everyone that needs it, the system must be low-cost and low-power.

2. The delivery of media content from the local server to the Android tablets must be able to be carried out without internet connection.

3. The system must be easy for users of all technical experience to interact with.

Chapter 3

Use Cases

3.1 User

The user may use the system in two ways. The first interaction with the system is to access and use the mobile application itself. They have the ability to initially access the 'About' Page, which provides a description about the application, why it was created, and its ultimate goal, the 'Help' Page, which allows a user to send a message to administration, and the 'Home' Page, which allows a user to select educational content. From the 'Home' Page a user selects their educational level, then the topic, then the subject, and finally the educational file to access. This can be observed in Figure 3.1.

The user may also interact with the Raspberry Pi, accessing it to turn it on and off, and to charge it. The wireless network is enabled automatically when no Internet is detected. Additionally, all Android application packages (APKs) should already be installed. As mentioned, on the application, the user will then be able to view files and read about the application, observable in Figure 3.2.

3.2 Administration

The administrator of the system may edit the system by creating new content for the application to display, updating the repository of data, and responding to User's messages submitted from the 'Help' Page. More detail can be seen in Figure 3.3.

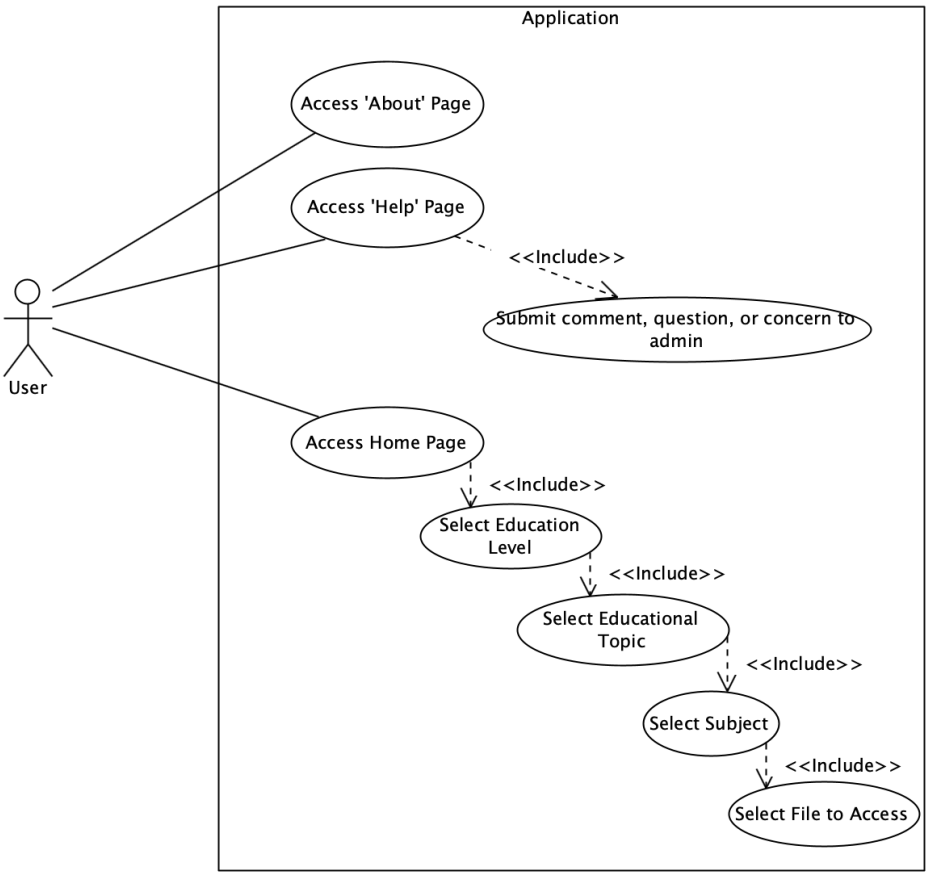

Figure 3.1: Use Case Diagram for the System's Application

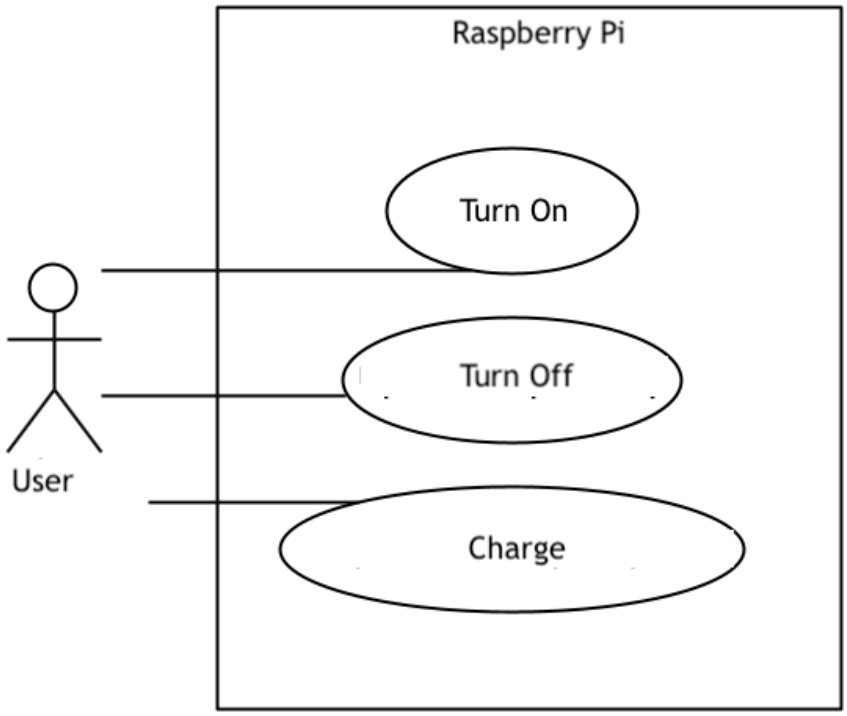

Figure 3.2: Use Case Diagram for the System's Raspberry Pi

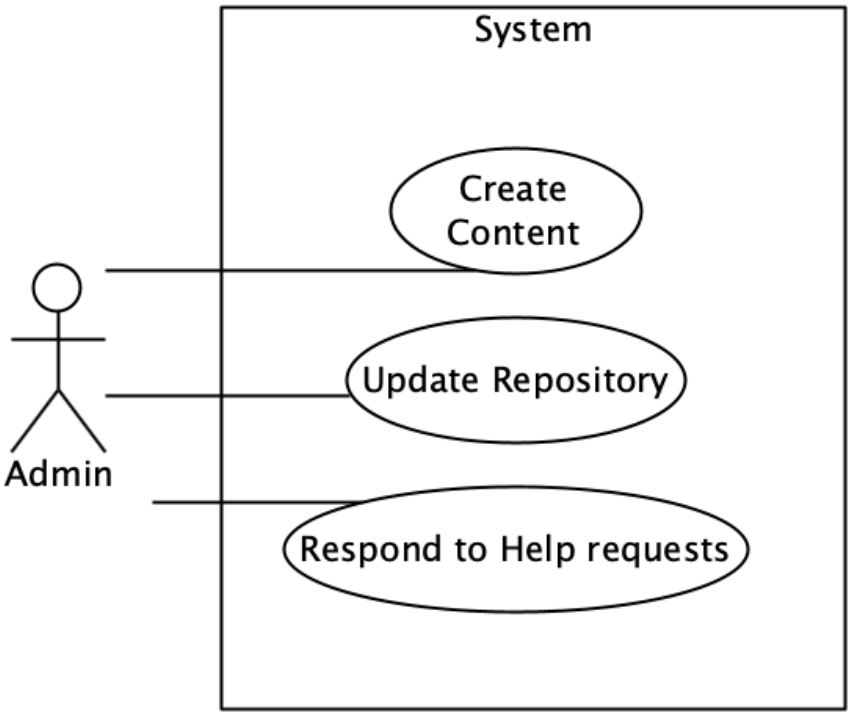

Figure 3.3: Use Case Diagram for the System's Administration

Chapter 4

Activity Diagram

4.1 Diagram

The activity diagram in Figure 4.1 outlines the flow of activities a user can complete when using our application. Upon opening up our application, MARTHA, a flash screen will appear while the content of the application loads. After that, the user is directed to the 'Home' Screen of the app. There will be a navigation bar on the bottom of the page so the user may click to navigate to the 'About' Page or 'Help' Page from the 'Home' Page.

The 'About' Page displays a text description of the application's mission and the inspiration for creating the application. Additional, information about the Victoria Relief Foundation, their mission, and their involvement in the creation and distribution of the application and its lesson content will be included. If the user clicks on the 'Help' Page icon they will be redirected to the 'Help' Page where text below text boxes will instruct a user to fill in their first and last name and message. At the bottom of the page is a button to submit feedback or ask for help. Upon clicking the button, a message will automatically be created that will be sent to the administrator.

If the user clicks on the 'Home' Page icon they will be directed to a screen that allows them to choose from multiple levels of lesson content, each increasingly more detailed than the last. This may also be referred to as the 'Lessons' Page. By clicking on a Level, a new page will appear that will allow a user to choose a Topic within that level, such as Vocational Education, Hygiene Education, etc. By choosing a certain Topic, the user will be directed to a certain screen where they must choose a Subject, such as Science, Math, English, etc. Finally, if a user selects a Subject they will be brought to a screen that displays each file pertaining to the Subject with a title to provide detail about the content of the file.

At any stage of file selection a user may navigate backwards towards the Lessons page by clicking the arrow in the upper-left-hand corner of the page. When the user clicks on a file they would like to view, they will be redirected to a full screen view of that file where they can read the file or return to the files screen. The user may close the app at any point in the flow of activities.

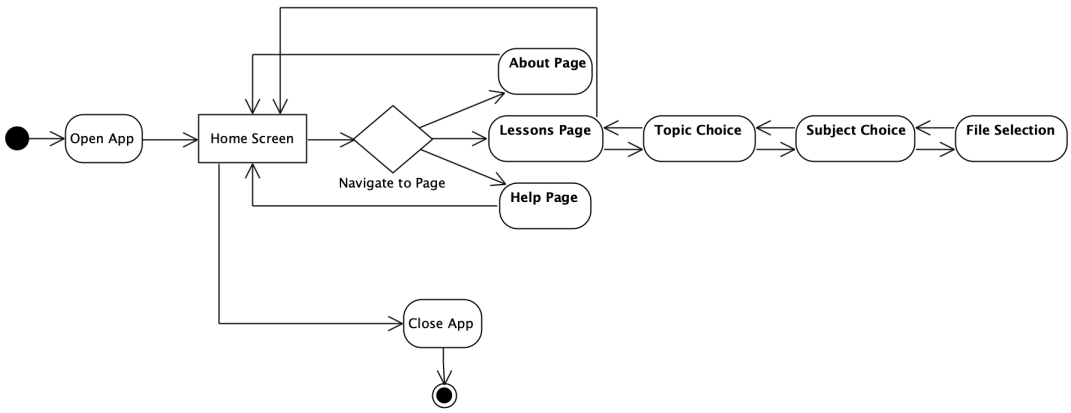

Figure 4.1: Activity Diagram for the System

Chapter 5

Technologies Used

In order to best meet the requirements, we identified several technologies and devices needed to implement our design. We decided to use React Native as our programming language, and solar-powered battery, a Raspberry Pi, memory card, Android tablet, and USB Drive as our main hardware.

Below is the complete list of devices and software we have used to develop our system:

- Android 10.1" tablet (Samsung - Galaxy Tab A (2019) - 10.1" - 128GB - Black) with Wi-Fi and Bluetooth functionality. This tablet runs on Android 7.0, and it will be able to install and run our app and have basic video playback functions and ability to view PDF files.

- Raspberry Pi 3B+/4B, 2GB or 4GB RAM with a micro SD card containing an image of the operating system. Raspberry Pi is the best solution on the market for our needs for a highly portable, general-purpose computer. For its tiny size, it supports 32GB or more on board storage, USB and HDMI ports, as well as wireless connections via Wi-Fi and Bluetooth 5, which are all crucial for our system.

- Raspbian OS. Raspbian is the official operating system of Raspberry Pi, which is a Linux(Debian)-based, low-resource operating system. It will maximize our development time and dedicate the most CPU processing time, RAM and storage space on our Raspberry Pi for the server processes.

- React Native. React Native is one of the most popular UI library for mobile development with a relatively flat learning curve. It has lots of easy-to-use development tools to choose from.

- Expo CLI. Expo CLI is a command line application that is the main interface between a developer and Expo tools. It can be used to create new projects, develop applications (running project servers, viewing logs, opening app simulators), publishing applications, and building binaries (apk and ipa files). It makes developing a project extremely simple and straightforward, especially given the thorough documentation, and it is compatible with React Native.

- MySQL. MySQL is extremely compatible with React Native coding environments and useful in storing data in an organized fashion, with clear labeling of metadata. It is specifically designed to make front-end development easier. It will be useful to store content and keep it updated in ways that will enable filtering and rendering on the front end.

- PHP. PHP is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. This makes it an ideal tool for communicating with the MySQL database.

- Strapi. Strapi is a headless content management system that acts primarily as a content repository. It utilizes JavaScript, which is particularly useful in setting it up since we are familiar with it. It allows for easy integration of GraphQL and SQLite using plug-ins. This will eventually streamline database management in the second phase of system development.

- GraphQL. GraphQL is a query language for APIs and a runtime for fulfilling those queries with existing data. It is naturally integrated as the schema for the database and its syntax is simple and easy to use with the React Native framework.

- SQLite. SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured SQL database engine. It allows for easy management and storage of our data.

- DigitalOcean Cloud Server. DigitalOcean is a simple server that works across multiple hardware devices in parallel. It will bring our server content closer to the user, reducing lag time.

Our goal was to keep the system scope and requirements simple to make the system easy to use. This has allowed us to focus on creating a robust and simple design. This also somewhat limited the variety of bugs and other technical issues we encountered during development.

Chapter 6

Architectural Diagram

6.1 Description

Our architecture is data-centric, rather than data-driven, with clients retrieving data that is then stored to their local device(s). There are two major parts of our system, which operate asynchronously. The first part of our system consist of an administrator uploading files to the MySQL database, and therefore the Santa Clara University file server, or Strapi database. (The database used varies depending on the phase of project development). The database will serve as persistent storage for our files that can be edited at any time, given the administrator has Internet connection. The schema for the database will be configured and accessed using PHP or GraphQL.

The second part of our system will consist of the Raspberry Pi server and local devices that will stream from it. The Raspberry Pi server requires modules to be installed and set up for it to function as a media server, as well as a script that will constantly run to connect to the database and retrieve data if Internet access permits it to. The Raspberry Pi will also provide an ad hoc wireless network in a 25m-30m range. Within this network, local devices may connect in order to download content that is stored on the Raspberry Pi server to the device by observing what is available and selecting desired content. The Raspberry Pi will allow this to happen by serving the media and accepting client connections, then streaming media data to all connected clients in the network.

The integration of the Raspberry Pi will occur during the second phase of system development. Initially, the front end will retrieve files cached on the Santa Clara University file server via a URL. In the second phase, we will phase out the MySQL database and phase in the Strapi repository and its additional tools. The Raspberry Pi will connect to the DigitalOcean Cloud Server to retrieve content. It will also, as mentioned above, host a local wireless network for local devices to retrieve content from.

The client will be an Android application running on the tablet. It will display a library of media and handle display of data – which will typically be PDF viewing. Note that it is likely that more data will be available on the server than can be stored on local devices, so it is necessary that clients are able to choose the content they download.

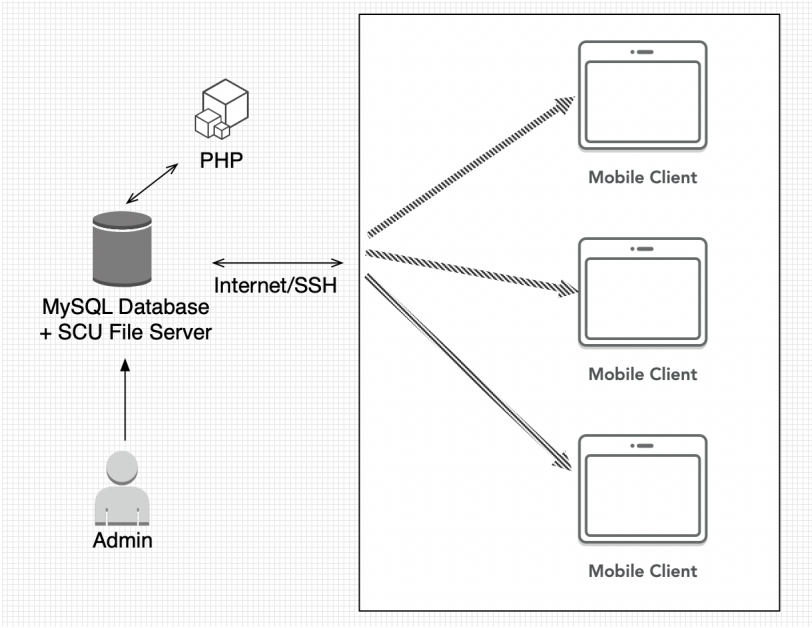

Figure 6.1: Phase 1: Architectural Diagram for the System

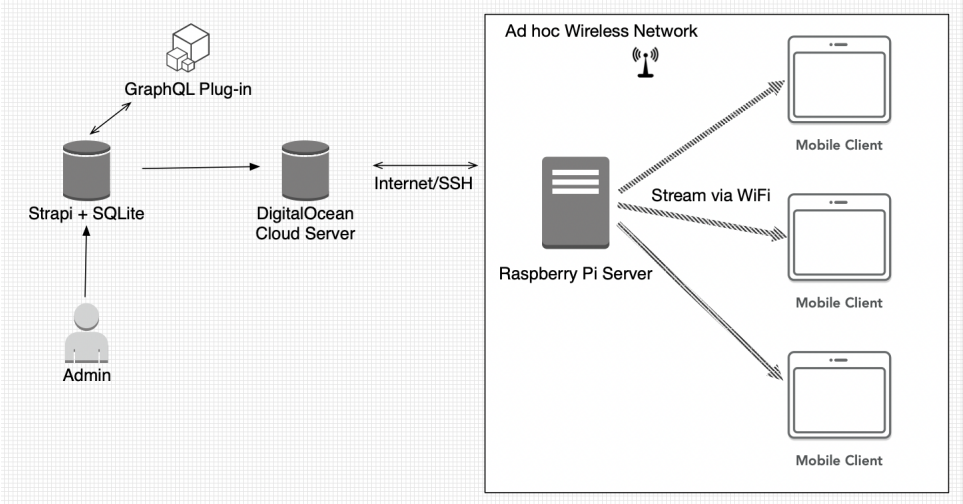

Figure 6.2: Phase 2: Architectural Diagram for the System

Chapter 7

Design Rationale

To meet the needs of the project, our group selected the following necessary devices and technologies to implement the design.

7.1 Design Goal 1

The system requires an on-site server and peripherals to deliver the data to local devices within the refugee camp. It must do this at a low-cost and with low-energy usage.

7.1.1 Goal

The physical devices we chose to include are:

- Raspberry Pi
- SD Card attached to the Raspberry Pi
- Android Tablet
- Raspbian (Operating System)

7.1.2 Rationale

The Raspberry Pi will act as the control hub that manages and is responsible for most of the functionality. The Raspberry Pi will not only store the education materials inside of the SD card, but it will also be responsible for hosting the ad-hoc wireless network that allows the Android tablets to connect whenever necessary. In addition, the Raspberry Pi only uses .4-3.4 watts/hour, requiring very little energy. One of the constraints is the lack of Internet connectivity in the refugee camps, and the Raspberry Pi is a compact device that can be easily carried to a location with Internet access to retrieve and download the latest data. After the Raspberry Pi is taken back to refugee camps, it hosts an ad-hoc wireless network that allows Android tablets to connect and download new data. The SD card enhances

the Raspberry Pi memory capability. The Raspbian Operating System will be responsible for managing the network protocol between the Raspberry Pi and the Android tablets, as well as conserving computing resources used by system processes to ensure most resources are dedicated to our application.

7.2 Design Goal 2

It is necessary to use languages that can easily construct a relay point to upload and manage data.

7.2.1 Goal

The coding languages used are:

- React Native
- JavaScript
- PHP

7.2.2 Rationale

React Native is a language that is extensive and popular and combines the functionality of React, JavaScript, CSS, and HTML to create a versatile and easy to use framework for building applications. It can easily be utilized to manage data and connections in our system. We will use JavaScript to manage the client and server side communication as a natural incorporation in the React Native framework. PHP will be used in intermediate configurations of the database and server.

7.3 Design Goal 3

The focus of the system is the mobile application responsible for displaying and managing data. Proper coding and management tools are required to do so.

7.3.1 Goal

Tools used to build the mobile application are:

- React Native
- MySQL Database
- Strapi
- DigitalOcean Cloud Server

- GraphQL schema plug-in
- SQLite

7.3.2 Rationale

React Native was used to develop and manage the cross-platform mobile application. It can easily be wielded to create an intuitive user interface given its wide range of developer tools and environments without the extra work of installing packages and installing environments from scratch.

MySQL is a simple database that allows for simple organization of database content and tagging of content to allow for simple access and sorting to the front-end. This allows for easy persistent storage and access of educational content from our front-end application. It was used as an intermediate database and server during phase one of system development.

Strapi contains open-source libraries and headless content management system that acts primarily as a content repository. It makes content accessible via an API for display on any device, without a built-in front-end or presentation layer and easily integrates several tools that streamlines content upload and retrieval. It will be integrated during phase two.

DigitalOcean Cloud Server is a server that runs in parallel and provides storage of content. It will be implemented closer to the user to reduce latency in data retrieval during phase two. It will pull content from the SQLite server that Strapi uses by default.

GraphQL is a query language for APIs and is a very simple way of interacting with AWS Amplify to manage data, both creating the schema and querying it. It is offered as a default schema when creating the AWS Amplify database. It enables the quick development of applications by giving software engineers the ability to query multiple databases, micro services, and APIs with a single GraphQL endpoint.

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. It is automatically used by Strapi and will act as an middle-end storage tool.

Chapter 8

Development Timeline

Our team generally followed the timeline below to complete our project in a timely fashion in order to meet expected deadlines from both Santa Clara University and our client, the Victoria Foundation. It is depicted as a general Gantt chart, seen in Table 8.1.

Legend	Team	Sarah	Leila	Marco	Deadline

Table 8.1: Development timeline arranged to conform to given deadlines.

	Fall Weeks 1-2	Fall Week 3	Fall Weeks 4-9	Fall Week 10	Winter Weeks 1-2	Winter Week 3	Winter Week 4	Winter Weeks 5-9	Winter Week 10	Spring Week 1	Spring Weeks 2-9	Spring Week 10
Problem Statement												
Design Report												
Design Review												
Revised Design Report												
Initial System Operation												
Application Front-End Prototype												
Server Set-Up												
API Calls and Back-End Flow												
Application Front-End												
Connecting Front-End and Back-End												
Admin Website Front-End												
Testing												
Final Presentation												
Final Report												
Final System												

Chapter 9

System Implementation

9.1 Overview

Our system is develop in two phases. During the first phase, the system consists of a MySQL database and an Android tablet with our application, MARTHA, installed on it. The front-end pulls directly from the MySQL database and downloads the content. During the second phase, the MySQL database is replaced with a Strapi content management system/repository in conjunction with a SQLite and DigitalOcean Cloud Server to manage data and store it persistently. A Raspberry Pi will be integrated to retrieve data from the DigitalOcean server when Internet access is available, acting as a local database, and host an ad hoc wireless network, acting as a local server, for the front end to connect to and download content from. The second phase aims to streamline the functionality provided in phase one, as well as reduce latency.

9.2 Raspberry Pi

We have chosen to use a Raspberry Pi in our system due to its versatility, price, size, and power usage. As mentioned, a Raspberry Pi is a computer that can be configured to do a variety of things. In this instance, we have configured it to connect to the Internet when the Internet becomes available and download new content from the MySQL database. When the Internet is not available, the Raspberry Pi will automatically generate an ad hoc wireless network for devices to connect to in order to receive educational content.

A Raspberry Pi allows us to provide an affordable, scalable, and maintainable system to communities that have very few resources, namely those living in refugee camps. The Raspberry Pi used in our system is only about thirty-five dollars. It will be used with multiple tablets and students, making the cost nearly negligible. For the same reason, the system scales very well.

Additionally, the system is small, and therefore easy to transport and store. The Raspberry Pi is 3.5 by 2.3 by 0.76 inches, and therefore a very portable option. The system is being developed for community with no access to technology or electricity, so must use as little power as possible. A Raspberry Pi only requires 3 amps power supply

which can easily be produced from a solar powered battery.

9.3 MySQL Database

A MySQL database persistently stores all files in the system. It provides an easy and functional way to organize files. It was easily integrated into both the front-end and Raspberry Pi. The application pulls content directly from the MySQL database providing a straightforward data path. It also does not incur monetary costs to the system as it is used. Overall, it is a simple and intuitive piece of the system.

9.4 Strapi and Additional Tools

A Strapi repository database persistently stores all files in the system, but provides simpler administrative system management and automatic integration of GraphQL and SQLite to configure and store data. DigitalOcean Cloud servers retrieve data from SQLite to store data closer to the user. Ultimately, the Raspberry Pi will pull updated content from the DigitalOcean server and serve as a local database.

Replacing MySQL with Strapi and its additional tools will make data management easier for administration and store data closer to the end user, improving system functionality, reducing latency, and increasing system robustness.

9.5 User Experience

One of the main accomplishments of the system is that the system is extremely simple and easy to use given that members of the community we intend to deploy to have never used electronic technology. The user interface design and user experience are critical to provide a system that is easy to navigate.

To achieve our goal, we organized our content hierarchically and by educational category and employed a minimalist design strategy. We used colors to emphasize different educational categories, simple font throughout, and offered simple work flows throughout. The content is organized by grade level and then educational category, as seen in Fig.9.1 and Fig.9.2. This design is intended to make our application simple, intuitive, and approachable.

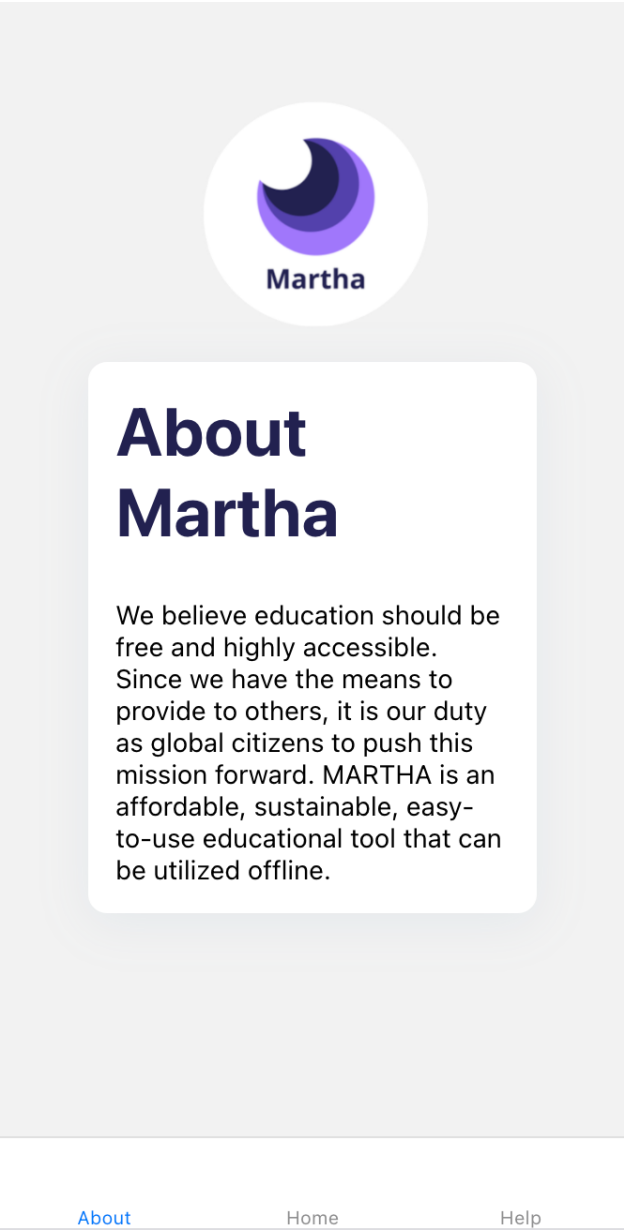

Figure 9.1: About Screen of the Application

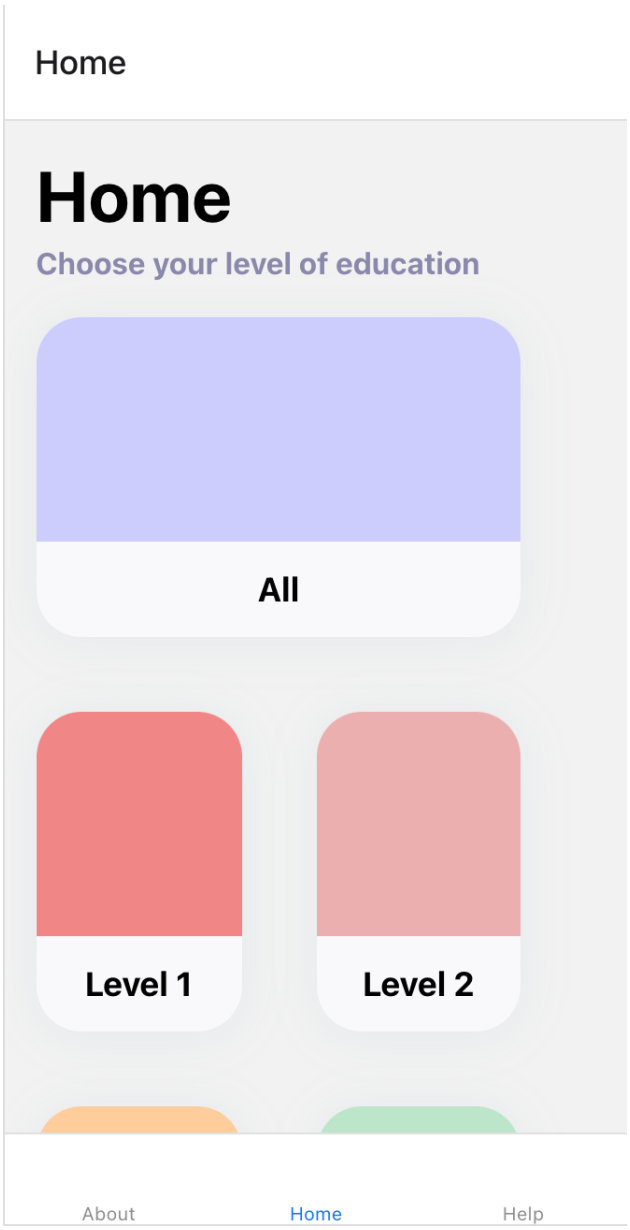

Figure 9.2: Home Screen of the Application

Chapter 10

Testing

10.1 Objective

The objective of testing is to detect and correct defects in the system as early as possible. More specifically, once defects are detected, they must be documented, prioritized, and the most critical should be fixed. Testing was conducted on individual pieces of the system, as well as the system as a whole, including the database, Raspberry Pi, and the front-end application.

The test plan prioritizes tests according to the system requirements. The hope is that the system will be deployed in the near future. We began our testing by focusing on unit testing, then integration testing, then system testing. The final steps of the process are closed-box testing and user acceptance testing.

10.2 Unit, Integration, and System Testing

As mentioned, the first phase of testing focused on meeting system requirements and testing using unit, system, and integration testing. The development team tested individual units mainly for functionality, but also for clarity of code. Unit testing prioritized the following:

1. The tablets will download content from an external database.
2. The Raspberry Pi will be configured to seamlessly and autonomously update its content by pulling from a larger online web database when it is able to connect to the Internet.
3. Sufficient educational content should be available without the need for Internet connection.
4. The system will be accessible and useful for people with limited to no technical experience.
5. The application user interface will be simple and intuitive.

In summary, we hope that content will be pulled from an external database, be loaded onto the Android tablet, and be easily accessible for all users.

At any point in development, there should be a path to upload content to a persistent database, whether that is Strapi or MySQL, and retrieve it on the front-end where it will be available to users without Internet connection. In the final phase, the Raspberry Pi should be able to store database content, automatically update itself, and automatically create an ad hoc network for external devices to connect to. The front-end application should provide an intuitive and simple interface for users to select educational material. The database should store educational content.

Each of these units were tested individually before shifting attention to integration testing to ensure parts of the system integrate seamlessly. Once unit and integration testing is completed, the system testing will begin. In system testing we will ensure that the application has a path to retrieve database content that can be viewed by users.

10.3 Deployment

The final phase of testing will focus on the ease-of-deployability and ease-of-use of the system. The system should be easy to deploy in the sense that any person could take our system to a school with no internet connection or electricity, and set up the system to download files to local tablets. The system should be easy to use for the users who are accessing the files. The wireless network should connect without user intervention. To test these parts of the system we hope to enact a beta test with students at Santa Clara University, then conduct user-acceptance testing within the refugee camps in Nigeria.

We will begin this process by asking peers to use our system and ask for feedback on the usability, intuitiveness, and performance of the application and system as a whole. The application as a whole, should be easy to use for a person who has had limited experience with technology. The interface should therefore be simple and straight-forward to use. In addition, the system must perform with little error and handle errors with ease.

By testing on local groups before conducting user-acceptance testing in Nigeria will ensure that basic errors will have been corrected before we bring the system to the community it is intended to be deployed in. Therefore, once brought to Nigeria, we may focus on correcting usability and deployment errors.

The final testing phase will be the user acceptance testing in Nigeria. The Victoria Relief Foundation will assist the Santa Clara University team in deploying the system to the refugee camps. Teachers and students in the refugee camps will test the system. Ideally users will be able to use the system without much instruction and maintenance. Our hope is that the Raspberry Pi is started with ease, the Android tablets connect to the Raspberry Pi seamlessly, and the users are able to easily navigate the application's interface. The team would like to see that the interface is simple, yet engaging and that the system is intuitive to use and continue to maintain. After this final testing, the team will make final changes and finalize the system. All testing results and iterations should be documented.

Chapter 11

Deployment Impacts

In this chapter we evaluate the various types of impacts our system will have on the local and global community once deployed. Important factors we consider are the risks associated with our product, ethical impacts, and societal impacts.

11.1 Risk Analysis

Our system presents minimal risk since no one experiences any additional harm when using our system, emotionally or physically. The information presented on our application has been crafted by students specifically informed about the most relevant educational topics and vetted by the Victoria Relief Foundation. This should ensure the content is both relevant and culturally appropriate, therefore minimizing or eradicating any emotional harm. Physical harm is limited to the narrow potential of a wire short circuiting and simultaneously being exposed to water, electrocuting anyone touching the device. This is highly unlikely since wires are largely insulted and unlikely to short circuit.

By providing detailed documentation of our system and its functions to users and members of the Victoria Relief Foundation, we should be able to provide thorough system understanding and avoid any potential risks from occurring. We also offer user training if necessary. By tailoring our system to client specifications, the system should be easy and simple to use while providing content that is useful.

11.2 Ethical Impacts

This application is intended to deliver educational content to a community that does not otherwise have adequate access to it. All humans should be given equitable access to information and educational tools. Education, and therefore opportunities that follow, should not be limited by geography or money. Since it is within our capabilities to create a solution that brings educational materials to refugee and impoverished communities, that is what we have done.

Our system is also economical, especially when compared to other alternatives. We provide the initial tablets and

server at no cost, and all content is designed by Santa Clara University students and volunteers. In addition, the system will be utilized by many students and teachers, making the costs of each device negligible. As mentioned, the system was free to develop and distribute.

Our system is sustainable since we have employed battery saving methods. We use a solar-powered battery to charge the tablets and to power the Raspberry Pi. We have also taken measures to reduce background computation in our system to reduce energy consumption further. Since the system is used by many users, it makes it more sustainable than systems used by only one person. This makes the system overall less environmentally taxing that other electronic systems.

The main negative ethical consideration is how accurately the content engages with users and delivers culturally appropriate information. While we have attempted to speak with those immersed in the refugee communities to capture the needs of our audience, we can only understand so much without actually experiencing the environment for ourselves. Our content is intended to educate users in traditional academic subjects, but also vocational training and emotional and physical wellbeing. If designed effectively, users should be able to easily navigate the application and learn. If we distract from the content or do not organize the content in a way that is easy to navigate there is the potential for causing more stress, denying people ease of access to educational material, or accidentally disrespecting others' values by displaying content incorrectly. However, we believe we have been able to avoid this issue through the help of the Victoria Relief Foundation.

11.3 Societal Impacts

The society most directly impacted by the deployment of our system is the Cameroonian refugee camps in Nigeria that the Victoria Relief Foundation has been supporting in the ways that they can. Our goal is to provide access to educational, vocational, and health information through our mobile application. Many aspects of our project ensure the societal impact is positive.

Our application is free to deploy and to use. Therefore, it is economically positive, preventing users from unnecessarily spending money on education and providing easier access to information. It also contains politically neutral content, ensuring it is a useful and positive resource. As mentioned, the content aids users in accessing educational and health care information, again providing a useful and positive community resource.

The platform is simple and easy to use and presents no security risks. The user interface is minimalistic and easy to navigate. No information is needed to log in and use the application, presenting no information security risks. The system is also easy to deploy and manufacturing it is easy in comparison to most technological systems.

It is worth mentioning the principle of informed consent. This principle relates to our project because the system has been designed for a specific community based on specifications from the Victoria Relief Foundation. The system

will not be deployed without community consent, therefore ensuring no one is at risk and understands the system and its functionality before using it.

Throughout the development of this project we reflected on the qualities that ”good” engineers possess and what their tools offer society. At the core of a good engineer is compassion for the community they are designing tools for. In addition, a good engineer has the foresight to plan for changes in user needs and plan for the future, ensuring the product they design is adaptable and sustainable. Throughout this project, we have attempted to consider the ways users will need to interact with our system to gain a benefit, limitations on system use, deployment effects of our system, and how user needs will evolve. By considering these factors we have ensured our product fits the current needs of Cameroonian refugees, can be used without Internet connection and other resources, can be updated with ease, and can move locations and be modified. We also ensured the product was sustainable by using solar-powered batteries and low-power devices and by allowing multiple users to use the system at one time. Our goal is to enact positive change, ensuring the world has more equitable opportunities available. We hope our system provides impoverished communities with little to no Internet access the ability to access educational, vocational, and health resources.

Chapter 12

Conclusion

To conclude, we will discuss obstacles encountered, lessons learned, and next steps for the project.

12.1 Obstacles Encountered

Our team encountered a wide range of obstacles throughout the course of this project. The largest obstacle we faced was the fact that we were all separated and working remotely due to protocols enacted due to the spread of Coronavirus-19 (COVID-19) around the globe. This severely hindered our ability to coordinate and integrate system units, as well as enact sufficient testing. Our solution was to communicate often and work to build a working path of data flow for our system in hopes that it will be further developed in the future. In addition to the strain of COVID-19, we lost a teammate after our first quarter, leading to some setbacks in the development of the Raspberry Pi unit of the system.

After gaining a new teammate, we then encountered difficulty in working with new technologies, most specifically the Raspberry Pi and supplementary database. We attempted to conduct thorough research, reach out to teams that had worked on similar projects before us, namely the YouLearn team, and read available documentation and forums that detailed information on the Raspberry Pi, creation of ad hoc wireless networks, file server implementation and passing file information.

Another challenge was lack of ability to fully perform integration and system testing due to the restrictions from COVID-19. As a result, we split our development into two phases. The main difference in the two phases is the different databases. In order to create an effective database we used two different methods, Strapi and MySQL, to ensure that we had a working database.

Debugging was another challenge faced. There were several instances where we would get stuck on one simple line of code or a single command. To resolve this issue, we usually did research on available forums and would return to a problem after a break, where we were able to more easily debug the system.

Throughout the process, constant communication and flexibility in goals allowed us to create a simple, yet working solution to the initial problem.

12.2 Lessons Learned

While developing MARTHA, our team learned the details of a software development life cycle. We learned how to communicate with stakeholders and glean user needs, document every step of the design process, iterate on initial design, coordinate with teammates and stakeholders, and be flexible with our goals in order to develop a working solution. Each step in the process allows one to create a product that can be easily and effectively used by others and further iterated upon to create something that is even more useful to communities around the world.

Due to remote working and the nature of software development, we learned how critical regular communication and coordination of goals is as well. We held weekly team meetings, regularly checked in with our advisor, and regularly checked in with other stakeholders. This allowed us to keep each other accountable, troubleshoot problems together, and progress project goals forward in a timely fashion.

Lastly, we learned the importance of flexibility in engineering projects. As one begins to design and test units, certain errors or issues might lead one to find a better solution to the problem at hand. In order to effectively create system units and integrate pieces together, we needed to communicate, conduct unit testing, and incrementally advance our project. Overall this steady testing allowed us to progress faster and more effectively.

12.3 Next Steps

Our system units meet all of the functional, non-functional, and constraints we initially set out to accomplish. Phase two still is incomplete. While the Raspberry Pi is able to successfully host an ad hoc wireless network that devices can connect to, it is not yet able to send files over the network. In addition, the Raspberry Pi and mobile application do not have automatically storage management. All other pieces of the system work as intended. In the future, we hope that the Raspberry Pi can successfully send files over its ad hoc wireless network and manage storage.

We also hope to be able to conduct our user testing and deployment testing in the future. This will allow us to better understand what gaps in technological understanding we have overlooked and how we may improve our system design and user interface.

We would also like to integrate a solar-powered battery that charges the Raspberry Pi and Android tablet. This would be a final step before deployment since it is only a matter of buying the battery. We would also like to change certain settings on the Raspberry Pi so that its operating system uses less power overall.

After these corrections to the system are made, creating a script that seamlessly sets up the Raspberry Pi would make the system much more scalable. It would allow Raspberry Pi boards to be set up quickly for use with our system, rather than taking a couple hours. These steps would make MARTHA easier to use and meet all system goals.

Bibliography

[1] “Rachel.” https://worldpossible.org/rachel.