Design Document

Mobile App for Forest Ecology Research

CS461 - First Term - Team Name : MAFE

Sergei Poliakov, Joseph Landreville, Moyez Ikhlas

November 2019

Abstract

The purpose of this document is to provide a detailed overview of the different components within our project and how they will be implemented. The document begins by defining the purpose and scope of the application, which we will be building. Following that, the document proceeds to describe any particular design concerns which need to be taken into consideration when implementing each particular component of the project. Each design concern is also addressed with how it will be handled.

CONTENTS

1 INTRODUCTION

1.1 Purpose

The purpose of our project is to help the Pacific Northwest Permanent Sample Plot Program transition away from their old digital data-collection toolset, towards a more modern solution using Open Data Kit X (ODK-X).

1.2 Scope

Since our goal is to help the PNW-PSP migrate services, the scope of the project is fairly large. We will be responsible for setting up the server and database software required to run ODK-X, customizing the user interface to fit our client's needs, and creating visualizations within the ODK-X Android apps to display the data collected by the apps in meaningful ways. As a stretch goal, we are also considering the creation of a service which simplifies the creation of the forms used by ODK-X to collect information from users.

1.3 Context

The PNW-PSP monitors a number of different forest plots in Oregon and Washington. Every year the PNW-PSP visits these plots and collects data on the trees in each one, which is later used for a number of different research purposes. The application which we will be developing, throughout the next two terms, will be responsible for facilitating all of the data collection, which will be performed by the PNW-PSP at the forest plots.

1.4 Summary

The application's design implementation is broken down into 3 specific areas: Data Management, General UI, and Visualizations. The description of each is as follows. Data managements is responsible for handling the data collection and data management aspects of the application. General UI is responsible for the implementation of a highly modernized and user friendly interface. Visualizations is responsible for displaying all of the pertinent application data in a clear, concise, and consistent manner.

2 REFERENCES

[1] "D3.js - data-driven documents," 2019.

[2] "Using odk tables - map view," *Open Data Kit Documentation*, 2017.

[3] "Download areas and navigate offline," *Google Maps Help*, 2019.

[4] "Using odk tables - navigate view," *Open Data Kit Documentation*, 2017.

3 GLOSSARY

Term	*Definition*
API	"Application Programming Interface". A set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.
MAFE	Mobile App for Forest Ecology
PNW-PSP	Pacific Northwest Permanent Sample Plot Program
ODK-X	The Open Data Kit X toolkit. An open source project which aims to create an accessible data collection platform.
ODK Survey	An Android app supplied by ODK-X which serves forms to the user to be filled out.
ODK Tables	An Android app supplied by ODK-X which can display information collected from the database in meaningful ways, as well as direct users to forms which will be launched in ODK Survey.
ODK Services	An Android app supplied by ODK-X which manages interactions with both the database which exists locally on the device, as well as the database which exists remotely on the server acting as a ODK Sync Endpoint.
ODK Sync Endpoint	Server software provided by ODK-X which talks to the various mobile devices running ODK Services in order to store the data they collect in a database running on the server, as well as sync data back to the mobile devices when requested.

4 DESIGN VIEWPOINTS

4.1 Data Management

4.1.1 Design Concerns

Data management aspect of this project embodies within itself two specific responsibilities, which are data collection and data management.

There are a number of different factors that must be addressed when implementing the data collection element into our application. This particular component is essentially responsible for fulfilling the applications primary purpose, therefore being extremely meticulous and observant during its implementation is vital. One of the most important parts of having good data collection in our software, is making sure that all of the forms for data input are easy and simple to understand for the user. Also it is important that our code is written clearly and is well documented, in order for our users to be able to make adjustments to any of the data input forms in the future, if needed.

A lot of the time when people will be using our software, they will not have any form of internet service, which means that their devices cannot be directly connected to the remote database. Therefore our software has to be capable of storing data on a local database and then sync the data with the remote database whenever the device gains network access.

When designing our database, it is extremely important that we choose a particular database model that is most optimal for the particular type of data which we will be working with. Choosing the right database design

model for our project is important because it will help us avoid common database issues such as ambiguous relationships and slow performance.

4.1.2 Design Implementation

The ODK framework provides us with a number of tools from its Tool Suite, which will help us implement a number of different features into our application. The particular tool from the Tool Suite that we will be using for implementing the data collection portion of our application is the Survey tool. The Survey tool is based on HTML, CSS, and JavaScript in order for designing forms for collecting data in our application. In order to make sure that our data collection forms our simple and easy to use, we will evaluate them using Jakob Nielsen's, renowned, 10 Usability Heuristics. In order to make sure that our code is easy to understand for future developers, we will also make sure that our software is deprived of any code smells.

We will be using the SQLite3 database for our project, which allows for local on device storage. This will allow the users to store data, without any internet access. In order to sync the data from local devices to the remote server, we will utilize the ODK's Services tool, which is capable of performing all of the necessary operations needed for syncing the local database with the remote server.

In order to have a high functioning and flexible database, we will be using the Relational Database Model for designing our database. We chose to use the Relational Database Model for three specific reasons. The first one being that it is known for its simplicity. This is of great importance to us, since the majority of the people who will be working with our database, in the future, will not be coming from technical backgrounds, therefore the less complexity in our data, the better. The second reason, is that the Relational Database Model is known for its data integrity. This is extremely important, since the database will be holding large amounts of data that has been accumulated over the past couple of decades. The last reason is its flexibility. The Relational Database Model provides a lot of room for extensions and scalability. As the project evolves over time, it will be convenient to have a database that can be easily modified when necessary.

4.2 General UI Design

The general UI of the application is going to be a deciding factor in whether or not our target users are going to want to use the app or not. There are three areas to focus on when designing the general UI.

4.2.1 Material Design

Google has an established system of applying colors to applications to help developers design the UI of their applications. This color system is called Material Design. It ensures that the final UI design is harmonious and pleasant to use.

Material Design divides the different colors involved in the UI into different levels. These include Primary, Secondary, Surface, background, and error colors. There are a few more, and each of these has primary and secondary color options that must be chosen.

Our application will be used by professionals and students in the field and therefore needs to have a simple, clean interface with a minimal learning curve and zero clutter. All the options available to the user should be

visible on the Home Page so that there is no confusion. It should also use a dull color scheme so that when it is viewed in the dark, it does not blind the user or is challenging to look at for an extended period.

It is for this reason that we will use a clean grey as the primary color with different variations of it, to distinguish the available elements. Our secondary color will be a dull green to highlight selected components.

4.2.2 Timeline

When starting off deciding upon the UI of the application, the first steps involve asking the client directly how he would want the application to look like as well as writing down the main features of the application (Data entry, Visualization, Settings, Syncing Data from Server). After getting down all the requirements, understanding how and when the application will be used, in what kinds of different environments (day, night, rainy, cloudy). The different color schemes that we want to use based on the Android material design guidelines. We will also look at other android applications and see what they did right and where they went wrong to make sure that we do not repeat those errors.

The second part of the design process involves creating very basic mockups and show the client to make sure that the layout we have imagined is the same as the one the client would like. All the different screens and pages that are going to be available to the user. We will then meet with the client, who will decide which layout is the best. After deciding the layout, our next step will be to create a prototype using one of the many available software. (Adobe XD or Framer are some options) Which would bring to life the design of the application with all the different colors involved? Some software even allows the user to click on specific options and visualize how the actual software would respond, a handy feature when trying to deciding the different changes to be made.

After meeting with the client and taking notes on what he thinks, we will work on taking into account the different things that our client has pointed out and incorporated those into the design of our application. Examples of changes that are often made in this step include choosing a different color scheme or moving the layout around — trying out different options to create an intuitive and user-friendly UI, while regularly emailing the client with different ideas and options and getting the approval to move forward. After getting a final prototype design, given enough time, user testing at this stage would be the ideal next step. This last check is done as a final test before the actual implementation begins, to make sure that future users of the application, are happy with how it looks. Often the above two steps are either overlooked and enough time is not spent on them, but they are crucial to ensure that changes don't have to be made later on.

The final phase involves implementing the final UI according to the prototype. The front end of the User Interface is created following the final prototype design. At the same time, the back end functionality is also implemented. Finally, the final phase will involve setting meetings with Users who will be using the application and taking note of any changes that need to be made before showing the client our final version.

4.3 Visualizations

4.3.1 Design Concerns

There are two main areas to consider when creating the visualizations: usability as a visualization of data, and usability within the context of the app itself.

Firstly, the visualizations need to be able to stand for themselves when it comes to meaningfulness and clarity. The users need to be able to easily tell what data is being visualized and how that is relevant to what they were doing. There is little point in creating visualizations nobody can understand. The goal here will then be to be consistent with presentation and structure. Line graphs should have a consistent look within the app, for example, and should share characteristics (heading placement, styling, etc.) with all other types of visualizations as to promote learn-ability.

Secondly, it is important to remember that these will be implemented in an app with unique constraints due to being used almost exclusively outside and frequently without the guarantee of internet connectivity. These have a greater impact on implementation - if we have data overlaid on a map then we need to make sure the map will load without an internet connection - but it can also influence style as they need to be easily readable outside.

Finally, the implementation of the visualizations should be done in a maintainable way such that visualizations can be created and modified quickly with minimal additional code written per visualization. We should try to avoid putting ourselves in a position where we have to refactor code when implementing new visualization types, for example.

4.3.2 Design Elements

The library we will be using to create visualizations is called D3 and describes itself as "a JavaScript library for manipulating documents based on data," with an "emphasis on web standards [that] gives you the full capabilities of modern browsers without tying yourself to a proprietary framework." [1] What this does for is us gives a simplified API which we can use to create and manipulate the various elements required to create a given visualization. There are many JavaScript graphing libraries we could use which can each create a wide array of different graph types, however it is unlikely we would use most of the available types in a given library and it would be significantly more difficult to change the appearance of the graphs as it would be if we simply made them ourselves in D3. Making the visualizations ourselves with D3 gives us significantly more flexibility which we can use to tailor what we create exactly to our client's needs.

Now, D3 does great for your standard graphs and charts, however for this project we would also like to have map views which display the locations of the various research plots as well as the user's GPS location. D3 can do maps fine (D3 map article) but it doesn't work as well when it doesn't have an internet connection. ODK-X has a built in map view [2] which uses an embedded version of Google Maps which is capable of downloading portions of its maps to be used offline [3] as well as displaying the user's GPS location. We will probably start by just using the map view (or perhaps the navigate view [4]), but we may arrive at some other solution based on feedback from our client.