Adiza Awwal
Project Final Report
Professor Gail Kaiser **Due: Monday, May 6, 2019**

Mobile development qualitative analysis: Low code application development compared to traditional mobile development

The scenario I investigated:

I would like to investigate the application development process for traditional programming in comparison to low-code development. This project idea comes from my midterm research paper that studies the maintainability of low-code software systems. Professor Kaiser made the valid criticism that my paper did not actually test maintainability. This is because I did not entirely encapsulate an entire development life cycle of a product. In order to truly evaluate maintainability of low-code platforms, one would need the same participants to return to their apps at a later period in time as well as see the process of adding a feature to a live product. Another critique was the concept that developers not originally involved in the product from the beginning would have to be tasked to modify apps developed by others as an additional step in assessing maintainability. Given the constraints of time as well as the scope the course, I have decided to continue to understand the process of developing applications using low-code, however, the focus is not directly on maintainability but, the overall experience using low-code in comparison to traditional methods.

How I investigated low-code mobile development compared to traditional mobile application development:

I implemented two similar mobile applications with a fair amount of complexity using traditional programming techniques and then implementing that same application using low-code development. Throughout the process of developing these two applications, I then documented the development process to address instances when one type of platform should be preferred over another.

More specifically in order to assess some ideas or concerns expressed in the original midterm paper, I documented the difficulty and utility of developing an application using low code and using an application built on low code. The project is more so an overview and qualitative assessment on the build process between two very similar mobile apps using very different tools. In the ‘Final Project Progress Report’, I initially entertained the idea of conducting user experience study between the two

applications from both developers and end user, however, I ultimately decided against this form of investigation. I decided against completing user studies mostly because my research centers the perspective of the developer rather than the end user. And finally, I wanted to ensure I address topics that would be useful for those in the prospective user community that would value this research. The prospective community for which this research would be valuable are mobile application developers with an inclination to begin developing using low-code in the future.

Data:

My mobile applications do not use an external data in any true sense. The low-code application developed using Microsoft Power Apps (https://powerapps.microsoft.com/en-us/), however, has the capability to access external information via connections with cloud services (e.g. Google Drive, OneDrive, DropBox, etc.) Accessing data through these cloud systems wasn't something that I determined to worth implementing in React and therefore, I decided against these methods. Initially, I had the intention to use a question generation API, such as Quillionz, in order to dynamically retrieve tests, but according to my research, Microsoft Power apps does not have API capabilities. The questions for both Employment Satisfaction Surveys are all predetermined.

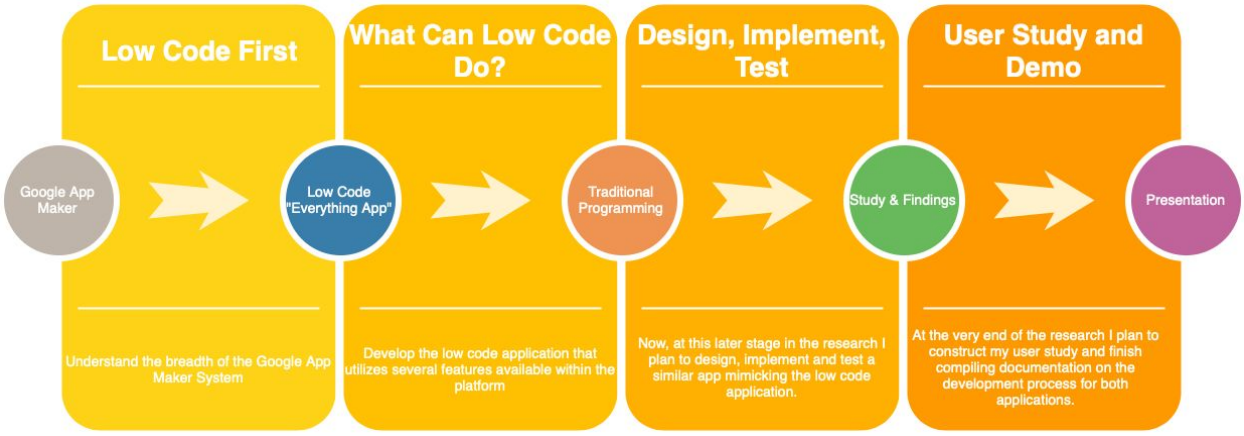
What's interesting about the system:

This concept of comparing low-code to traditional programming should address the three main criteria for the project: software developer productivity, software product improvement, and/or novel use cases for software engineering techniques, practices or concepts. This is forward facing research in that there are a lot of unknowns surrounding low-code and/if low-code should or should not be preferred over traditional techniques. The study can also address when low-code or traditional programming should be preferred over one another.

Additionally, I think this is a novel idea just because there are a lot of private companies which make significant claims about low code development but, there is little to no existing research on the limitations for low code or justification/criteria for selecting low-code over traditional methods. Ultimately that is the biggest question surrounding low code as well as the scalability of the low code development. For this particular project, it is important to understand the limitations and best practices for the low code development (e.g. the best cases in which low code can be preferred to traditional software development and vice versa.)

The application itself:

I began creating the low-code application first. Professor Kaiser suggested working with the low-code first as doing the reverse could foster issues with limitations on the low-code platform (e.g. building features low-code development environment does not support.) Initially, I first planned to build as many features available in the low-code development environment first and then complete the traditional application using traditional systems. The app was intended to be an “everything” app - utilizing as many features within low-code as possible - without necessarily making cohesive sense. This, however, proved to extremely difficult. As will be discussed for later in the final report, low-code platforms are *specific* yet verbose in their set intention. Because of this quality within the low-code development environment, it proved difficult for me to select an application to create without a set purpose or scope.



The ideation process for the application itself was quite time-consuming. The first step was deciding on which low-code application environment would be a correct fit. There were several choices and initially, I decide to use Google’s App Maker, however, I ultimately used Microsoft’s PowerApps. I decided to use Microsoft’s PowerApps because it is a platform marketed towards enterprise rather and a standalone tool. Microsoft PowersApps is designed to create internal business applications for corporate settings. Given most low-code is marketed or intended to expedite business-related task, I decided to use PowerApps for this reason as well as design the intention of the application around this theme. Both applications are designed as Employee Engagement Surveys with the concept to collect employee feedback on their employment satisfaction. In terms of the tools and systems for the traditional development, I decided to utilized React Native as I have had exposure in Advance Software Engineering COMS 3156.

For my final version of the application, I created an “Employment Satisfaction and Retainment Review” or survey. The user is expected to toggle through each screen and fill out the perspective on their company. The first screen is an explanation of the survey, the following five screens prompt the user with a series of questions, the second to last screen displays an informative video and the last screen thanks the user for participating.

Unexpected Challenges:

One unforeseen challenge developing this project was encountering how I would share my low-code application with Professor Kaiser and Ashna. This was by far an issue I did not expect given the sharing capabilities for this online application appears to be rather simple. I encountered a few hiccups in attempting to share the project with COMS6156 instructors via their own PowerApps accounts. There were several issues around permissions and data access via OneDrive and Google Drive, however, the situation was resolved after a few Piazza exchanges and troubleshooting in class with Ashna.

Team:

I’m working alone on this project, however, I consulted Professor Kaiser and Ashna as needed.

Research Questions and Assessment:

The benchmarks and research questions I chose to investigate are constructed from common traditional programming techniques and best practices. Previous development experience informed my selection and focus of research questions and criteria.

1) What are the limitations of low code platforms?
2) In what instances should low code be preferred over traditional techniques?
3) How does low code impact the life cycle of software development?

Because this study is qualitative, I think will only be able to compare and contrast between traditional software engineering precedence and what reveals itself using low-code.

Prospective User Community:

I think this research would be helpful for enterprise or organizations that are exploring developing some type of application but, do not necessarily understand the best methodology for their purposes. This research is helpful to understand the gradient distinction between low code and traditional development.

Areas to document while developing:

As pointed out by Professor Kaiser, all topics surrounding the applications have to be discussed for the research portion. Testing, maintenance, version control, development methodology, etc. are all portions of the project that will be reviewed and compared. I plan reference COMS 4156 testing material.

The following bullet points will only reference low-code development and assume familiarity with traditional software development as a point of contrast

- Time
- Ease of Learning / Adaptability
- Version Control
- Cross-Platform Apps
- Unit Testing / Testing

I found that low-code displayed its diversion from traditional programming in the bulleted categories above. First, time is a large factor in the need/desire for low-code application development. Low-code development is significantly faster than traditional programming even with inexperience in the low-code development environment. Another reason one may select low-code development over traditional would be the ease to learn and adapt to a new platform. Low-code's drag and drop interface make it both intuitive and easy for the end user no matter their skill level with traditional application development. The next and final component would be the version control system. With Microsoft PowerApps there is the ability to hard save individual versions of the application. Though all work on the app temporarily saves one when there is an initial 'save version' does the platform consider that a version of that can be reverted to. The version control system is a little messy in that there does not appear to be a way to access the intermediate version of the application. Another added functionality using low-code that is that it tends to be platform agnostic. Once one builds an application, that same app can be repurposed for either a tablet or mobile application on either iOS or Android. Though React Native generally works for development on both Android and iOS there do exist limitations. There also appears to not be a way to unit test anything which somewhat makes sense in that most things are hidden in the backend.

User Study and Demo:

As stated above, I decided against conducting a user study for two reasons. One is that I saw no obvious benefit to the potential community that would value this

particular research. The second I became constrained in time and did have enough access to develop and manage a user study.

Also for the demo linked below, I walk through both applications highlighting the nuances and distinctions between both in the code and the application itself. Both applications are similar to one another but, not identical.

What Adiza learned:

From this project, I learned several new and interesting concepts I don't believe I would have had to opportunity to learn in different classes. Firstly, the range and scope of low-code development platforms I knew nothing about prior to this project. After conducting this research, I feel I have the understanding to determine the best instances in which building from the ground up would be preferred over utilizing higher level resources. Secondly, I learned learning high-level tools such as Microsoft PowerApps is in itself a skill. Though low-code is a level of abstraction from normal programming, working with the interface in itself takes time and familiarity to become particularly good at. Lastly, I would say I increased my exposure and familiarity with the React Native framework.

Conclusions:

In conclusion, low-code development and traditional systems both serve a particular purpose. In the instance of low-code, this is a development methodology that should be used assuming that the desired application is something somewhat familiar, simple or commonplace. The low-code favors ideas that are already introduced in a standardized and conventional format. Traditional application development provides more availability for unique and customizable implementations. Low-code development can also be favored in the instance where the speed of development is particularly crucial. The time required to create a low-code application even with little experience is significantly less than that required of a traditional application. Also, low-code interrupts the development life cycle in that it not only shortens but, hinders the collective development process. I cannot forsee an entire team working to build a low-code application. I would imagine only individuals or small group would do so.

Project Demo:

This is the link to my final project video demonstration. I did not experience any issues with my demo pertaining to the applications themselves.
https://vimeo.com/333589190

Code Delivery:

All project deliverables and necessary information to run and view the React Native application is in the following repository:
https://github.com/adiza/coms6156

Documentation so that someone outside this class could 1. run anything executable, and 2. further develop any software produced can be found in the README.md file contained in my GitHub repository

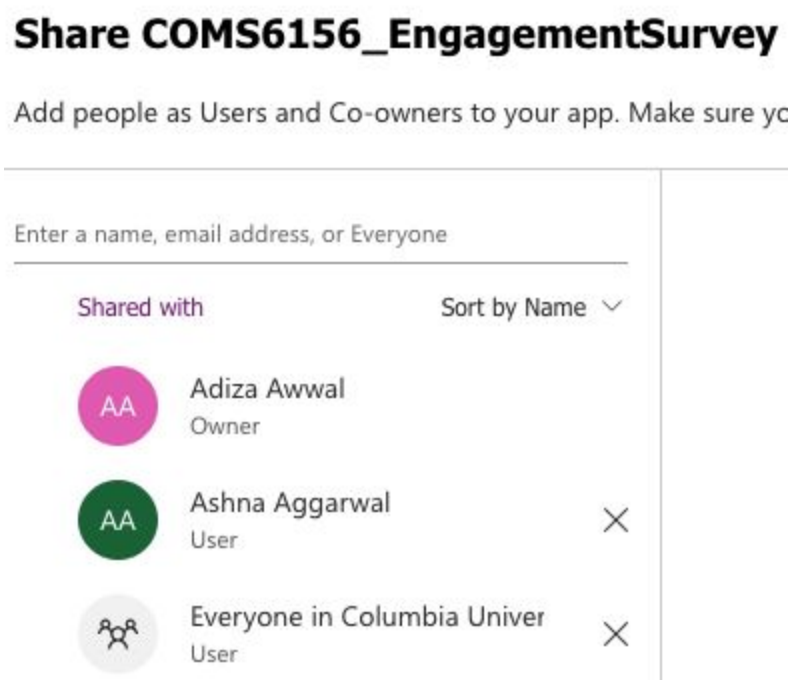
Also, all milestone assignments are contained within the **"milestone_assignments"** directory in the above repository (i.e. preliminary proposal, revised proposal, progress report, final report)

All open-source code referenced for React-Native Application:
In order to build my Employment Satisification survey, I used a package called react-native-simple-survey. This allowed me to build a survey screen using a JSON object storing the questions. I used the creator's ExampleApp as a template and reference.
Documentation:
https://www.npmjs.com/package/react-native-simple-survey
Creator's Github Repository:
https://github.com/devlinb/react-native-simple-survey

Low-Code Application Access:

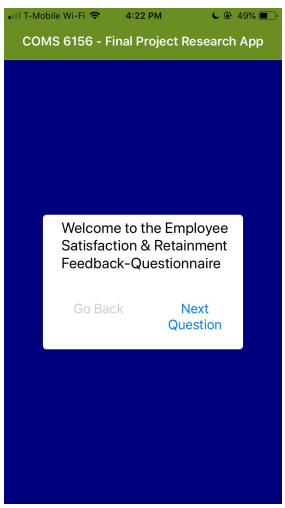
Ashna should have access to my low-code application. She should have user access and be able to access the application through her student portal logging in through her Columbia email.

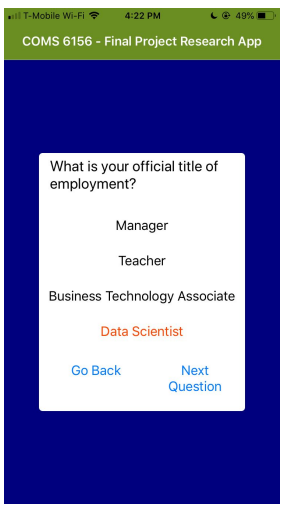
Once logged in, I Ashna should be able to view the project once she logs in or through this sharable link: https://web.powerapps.com/apps/ba7425ca-1cf1-4ac1-a9b3-2d6cc16711e0

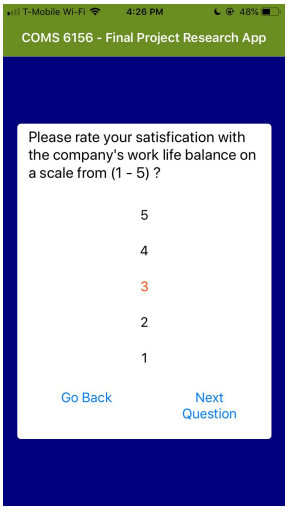


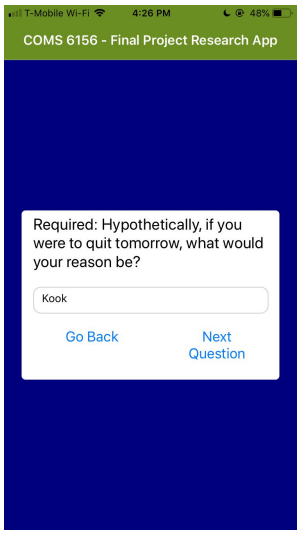
Building in React Native:
In order to make my app, I used Expo instead of an emulator (i.e. Xcode)
https://expo.io/

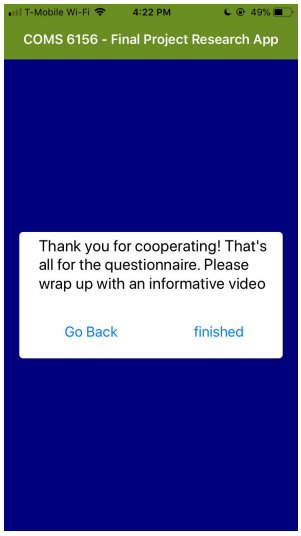
React Native Application Screens:

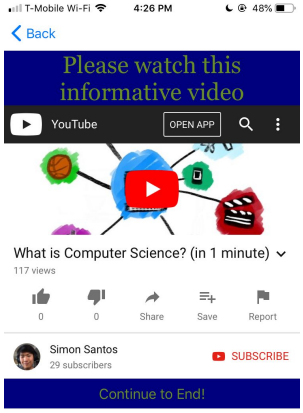


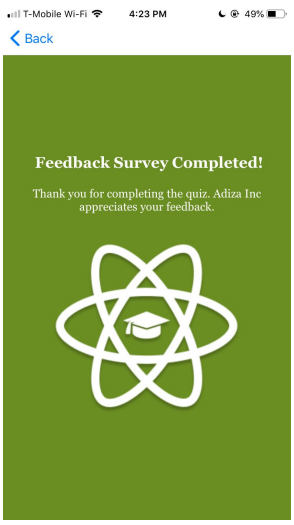












Low-Code Application via Microsoft PowerApps: