ODK 1.5: Implementing ODK Collect in 2024

Ariel Fu*
University of Washington
Seattle, Washington, USA
arielfu@uw.edu

Mitchell Ly*
University of Washington
Seattle, Washington, USA
lymit000@uw.edu

Annalisa Mueller-Eberstein*
University of Washington
Seattle, Washington, USA
anname37@uw.edu

Aidan Petta*
University of Washington
Seattle, Washington, USA
aidanjp8@uw.edu

ABSTRACT

In the realm of global health, Information and Computing Technologies (ICT) have emerged as powerful tools for improving health care delivery in low-resource settings. One notable example is the Open Data Kit (ODK), a free and open-source software designed to facilitate data collection and management in challenging environments. Despite its success, ODK is built on technologies from 2007, limiting its potential in today's technological landscape. This paper explores the modernization of ODK Collect as ODK 1.5 using Flutter, a contemporary framework. The new implementation aims to enhance functionality, streamline usability, and expand the platform's impact on health care delivery. Key improvements include a more user-friendly interface, cross-platform compatibility, and direct integration with Excel for data collection, bypassing the need for XML conversion. The project leverages Firebase for secure, scalable cloud storage, ensuring data accessibility and safety. This modernization not only preserves the core strengths of ODK but also adapts it to the needs of the modern world, offering a robust, versatile solution for digital data collection in resource-constrained environments. Future work can explore additional features like bi-directional communication, web-based options, and advanced data inputs to further enhance the platform's capabilities.

KEYWORDS

Mobile Health, Offline, Forms Collection, ODK, Flutter

ACM Reference Format:
Ariel Fu, Mitchell Ly, Annalisa Mueller-Eberstein, and Aidan Petta. 2024. ODK 1.5: Implementing ODK Collect in 2024. In . ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

*All authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In the realm of global health, Information and Computing Technologies (ICT) have emerged as powerful tools for development, particularly in improving health care delivery in low-resource settings. The integration of ICT in health care through mobile health (mHealth), leverages low-cost mobile devices to support health care providers and improve patient outcomes. Health care in low-resource settings often suffers from limited access to medical professionals, inadequate infrastructure, and insufficient medical supplies [2]. By providing tools for better data management, communication, and decision-making, ICT helps bridge gaps in health care infrastructure and access, especially in areas where traditional resources are scarce. These tools enable healthcare providers to collect, manage, and analyze patient data efficiently, leading to improved diagnosis and treatment outcomes. Moreover, ICT facilitates remote consultations and tele-medicine services, allowing healthcare professionals to reach patients in remote or under-served areas. This approach is not only innovative but also essential in addressing the unique challenges faced by health care systems in resource-constrained environments.

Digital data collection in poorly connected environments remains a significant challenge, especially in resource-constrained settings. Reliable and simple platforms for data collection are essential to ensure that critical information can be gathered and analyzed effectively. Open Data Kit (ODK), a free and open-source software released in 2008, has been instrumental in addressing this need by providing a robust solution for data collection, particularly in environments with limited connectivity [9]. ODK is designed to facilitate data collection and management in challenging environments. The "Open Data Kit project aims to empower resource-constrained organizations to build information services in under-resourced contexts through the creation of an extensible suite of open-source tools." [5] ODK has since empowered health care providers, researchers, and organizations worldwide to gather, analyze, and act on crucial health data efficiently.

Since its inception, ODK has played a pivotal role in various initiatives worldwide. It has facilitated monitoring efforts in the Amazon rainforest by members of the Surui tribe and the Brazilian Forest Service, enabled the Jane Goodall Institute to track conservation endeavors in Tanzania, supported The Carter Center in observing elections in the Democratic Republic of Congo, and even aided an astronaut on the International Space Station in monitoring

the progress of the Carbon for Water program [8]. As the user community expanded, so did the suite of tools, including ODK Build, a drag-and-drop form designer, and ODK Briefcase, a form management tool. The ODK website has garnered traffic from nearly every country, and it is estimated that millions of users globally have utilized ODK or its derivatives to date.

As technology evolves, so too must the tools we rely on to ensure they remain effective and relevant. Reimplementing ODK with today's advanced technologies offers an opportunity to enhance its capabilities, streamline its usability, and expand its impact on health care delivery in low-resource settings. This paper will explore the need for ODK as well as its modernization of ODK Collect as ODK 1.5. Then, detail the overall solution and implementation before finishing with reflections on the process and suggestions for future work.

2 PROBLEM DESCRIPTION

The original goal of ODK was to "provide a configurable set of open-source software tools for organizations to use in resource-constrained contexts." [5] These tools were designed to be flexible, modular, and offline capable. And the original product has certainly had an impact and filled a need.ODK has increased efficiency, reduced cost, and produced more accurate data.

ODK was designed to be a reliable and straightforward platform tailored for resource-constrained environments. It follows a three-step process: Build → Collect → Aggregate. This process has been applied in various contexts, such as rainforest management, tracking malaria cases, and personnel management. Since its inception, ODK has facilitated over 250 million submissions annually, showcasing its widespread adoption and utility [2]. The original technology relied on forms-based data collection where data is entered based on individual forms, which are then stored as records in a database. For example, in tracking a malaria outbreak in remote villages, health workers can collect data in the field using forms, which are later aggregated to monitor the spread of the disease.

Despite its success, ODK is built on technologies from 2007, which limits its potential in today's technological landscape. Reimplementing ODK Collect as ODK 1.5 using a modern framework such as Flutter can address several limitations and enhance its functionality. Modernizing ODK involves improving its layout and design using contemporary design principles to create a more user-friendly and intuitive interface. Additionally, cross-platform compatibility is crucial, developing a solution that works seamlessly across different operating systems (iOS, Android, etc.), making it accessible to a broader range of users. Expressing forms in modern Excel formats allows for easier creation and management of forms, simplifying the process for users. This also means avoiding the issues with Javarosa highlighted by the two creators consulted.

Our project proposal aims to develop a comprehensive healthcare data collection and management system that addresses critical challenges faced in resource-constrained environments. The proposed system will prioritize several key features to enhance functionality and usability. We will focus on strengthening security protocols to safeguard sensitive patient data against emerging cyber threats. The system will also offer multi-platform support, allowing healthcare providers to access and use the application across different devices and operating systems, with or without an internet connection. This flexibility is crucial for ensuring seamless data collection and management in remote or underserved areas where connectivity may be limited. Furthermore, the user interface (UI) will be designed with a clean and intuitive layout, facilitating easy navigation for healthcare professionals of varying technical backgrounds. Accessibility features will also be incorporated to ensure that the system is usable by individuals with disabilities. In terms of data ingestion, the system will support retrieval from Excel files for various data types including text, multiple-choice, multi-select, and time and date fields. This feature will streamline the process of importing existing data into the system and enable efficient data entry during fieldwork. The updates to the core framework will also support evolving data collection needs. This will involve enhancing existing features and incorporating new functionalities to meet the evolving requirements of healthcare data collection in low-resource settings.

Re-implementing ODK with modern technologies such as Flutter offers the potential to significantly enhance its functionality, usability, and reach. By addressing the limitations of the original 2007 technologies, we can create a more robust and versatile platform that better serves the needs of resource-constrained environments, ultimately improving the efficiency and effectiveness of digital data collection in poorly connected settings.The forms used for data collection would be designed in Excel, allowing for easy updates and modifications. The collected data would be synchronized with a backend service, ensuring that it is securely stored and accessible for analysis. By utilizing Flutter, the app would work seamlessly on both Android and iOS devices, providing a consistent user experience regardless of the platform.

3 RELATED WORK

The use of ODK and ODK-X suite tools is prevalent in medical investigations, significantly enhancing health programs and systems globally, but especially in regions of a developing country [2]. Particularly in nine countries including Kenya, Mali, India, Nigeria, Ethiopia, Madagascar, Tanzania, Mozambique, and the Dominican Republic, ODK has facilitated rapid data collection and analysis [9]. The success of ODK tools in these countries can be attributed to several factors: its open-source nature, offline data collection capabilities, submission via SMS, and ease of customization without extensive technical expertise.

3.1 ODK Applications

Recently, in 2018, ODK was integral to Somalia's Federal Ministry of Health and their mission to immunize 726,000 young children against the poliovirus [8]. This initiative was prompted by the detection of the virus in sewage samples from the nearby Mogadishu city, despite no reported cases of polio at the campaign's outset. In response to the urgent need for efficient data collection and reporting during the campaign, over 200 personnel were trained ODK, an open-source mobile data collection tool. ODK played a crucial role in facilitating the gathering of timely and accurate information, empowering public health officials to monitor progress and make informed decisions as the campaign unfolded. This case highlights the indispensable contribution of technology, exemplified by ODK, in supporting public health initiatives, particularly in regions facing

challenges such as political instability and inadequate infrastructure. By leveraging tools like ODK, health authorities can effectively combat infectious diseases like polio, ultimately advancing the global effort to eradicate such threats to public health.

The Humanitarian OpenStreetMap Team (HOT), leverages ODK to support a global community in mapping areas vulnerable to disaster and poverty. This mapping is crucial for humanitarian aid and sustainable development. Paper forms weren't working due to frequent flooding and HOT turned to ODK. "Data collection [with ODK] was much easier and much more effective" due to its flexibility and offline-first functionality [7]. Volunteers can easily use the ODK app to map critical infrastructure like buildings and roads, even without internet access. Offline satellite imagery can be included to provide essential context for disaster response.

In regions where reliable Internet is scarce, ODK proves indispensable for offline data capture. Historically, MORU (Mahidol Oxford Research Unit) struggled with paper forms, which were prone to transcription errors that were often discovered too late to correct [6]. ODK filled in that gap. ODK is optimized for offline data collection, offers industry-leading data protection and security, and ensures better data quality, which is crucial for informed public health decisions. Additionally, ODK's ability to log every action during data collection and management ensures robust data provenance, an essential aspect of clinical research. Its seamless integration with R and Python simplifies reporting by connecting to existing analysis pipelines. Beyond the software's technical advantages, Naomi emphasizes the supportive ODK community of over 14,000 members who share their expertise and provide mutual assistance. ODK excels in facilitating the collection of trustworthy data. This makes ODK an invaluable tool that MORU relies on for its research endeavors, ensuring the data collected is both accurate and dependable for the broader research community.

3.2 ODK Extensions

As the ODK community developed many different extensions of ODK, there were many different extensions of ODK that were published in research literature. One extension of ODK focused on adding features to ODK, one of which was deprecated in ODK when the Google Play Store began enforcing high-risk data restrictions. In 2019, there were 3,873 articles obtained from Google Scholar alone that mentioned that included the following keywords: Open, Data, Kit, ODK (and extension, architecture, workflow, and visualization) [9]. Below are three examples of ODK extensions: GeoODK, ODK-Ext, and Mezuri.

One example is GeoODK, an extension of ODK that focuses on collecting and storing geo-reference information. Similar to ODK, GeoODK also collected survey-based data in offline environments with a larger emphasis on mapping. The extension of ODK boasts additional mapping capabilities, collecting point data, converting collected data into spatial formats, preparing the dataset for a Geographic Information System (GIS), etc. One case study looked at road network development in near real-time in Dehradun, India. As part of the shift from traditional paper-based maps to digital formats, GeoODK is utilized for mapping small streets inaccessible through mainstream mapping platforms and ODK Build is used for form creation.This project resulted in the development of a comprehensive road network map, aiding in route identification and connectivity assessment. [3]

ODK-Ext is built off ODK with pre-filling forms based on previously collected data and also the ability to submit data via SMS [9]. For instance, if a relief organization is conducting health assessments, they can use ODK-Ext to pre-fill certain sections of the assessment form with demographic information gathered during initial registrations. This streamlines the data collection process, reduces duplication of effort, and ensures consistency across assessments. Additionally, ODK-Ext allows for data submission via SMS, enabling enumerators to transmit assessment data even when internet access is unavailable. This ensures that critical information is captured and transmitted in real-time, facilitating timely decision-making and resource allocation. Overall, ODK-Ext enhances the effectiveness and efficiency of data collection efforts in humanitarian contexts, contributing to more informed and targeted interventions in crisis-affected communities.

One more extension, Mezuri, was both a hardware and software system that was designed for complete end-to-end management of data [9]. Mezuri was a cloud-based data management platform that supported the entire data pipeline, from data collection to data visualization and statistics. For example, Mezuri streamlines the process of setting up backend infrastructure for data storage and processing, which is often a cumbersome task for users of ODK. By leveraging modern technologies such as Docker, Mezuri enables users to seamlessly deploy and manage their data processing workflows, reducing the time and effort required to get started with data analysis [4]. In addition, Mezuri places a strong emphasis on data transparency and integrity to reduce costs and errors while enabling better measurement of impact. One example application was cookstove monitoring in Darfur, Sudan–a study comparing objective cookstove adoption measured by sensors versus user-reported adoption measured by surveys [4].

4 OVERALL SOLUTION AND APPROACH

ODK, a set of tools designed in the early 2000s to build data collection forms and manage collected data in resource-constrained environments, is highly successful, even in the modern world. We began imagining how ODK would be rebuilt in the modern world by researching current use cases, the evolution of technical literacy, and modern development frameworks. In figure 1, there is a timeline representing our development starting in early April and culminating with a minimum viable product in early June.

4.1 Research and Design

The first step in the research process involved an in-depth investigation of the use case for data collection in resource-constrained environments. We began investigation by finding where ODK is being used and expanded into other research fields that involve large data collection and aggregation. This investigation revealed a significant increase in the use of mobile technology for data collection, even in the most remote areas. Moreover, in the past, much of the mobile technology revolved around Android devices. From our chats with healthcare researchers in resource-constrained environments and developers of ODK, we are seeing more usage of iOS

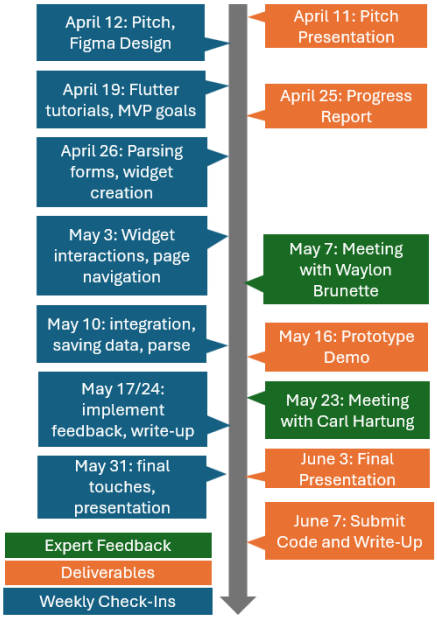

Figure 1: Timeline of research for and development of ODK 1.5. Diagram by authors.

devices, compared to when ODK was developed in the early 2000s. This shift can be attributed to the increased accessibility of mobile devices and general globalization. This research led us to consider modern frameworks that would support multiple mobile devices, as well as the web.

The next step was to understand how technical literacy has changed in the real world. With the proliferation of smartphones and the internet, people are becoming more tech-savvy. When we chatted with the developers of ODK, one of the struggles of building ODK was that many people were still collecting and aggregating data on paper. Then, once people turned to technology for data collection and aggregation, there were very few ways to represent the data. As a result, ODK had to keep building workarounds and adding features to support different data collection methods as technology evolved. For our solution, we also researched the most common tools in research for data collection and aggregation in the modern world, one of which remains Microsoft's Excel [1]. Microsoft Excel was also a software service that required additional features to be supported in the original ODK. For our solution in the capstone, we chose Microsoft Excel as our primary method for data collection.

As we chose a framework, we moved to the design process. One of our priorities for ODK 1.5 is an accessible and user-friendly design. Due to the nature of the capstone class, we are constrained on time and opted for a simpler design. We created a small and low fidelity mockup to align on the overall concept as a team and outline the workflow. The mockup helped us conceptualize the user interface and the interaction between different components of the application. It also helped us convey our design and provide a

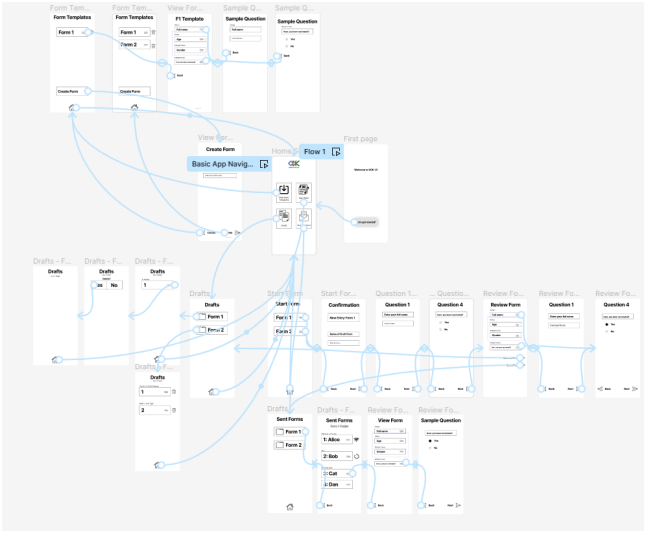

Figure 2: Model of entire system. Diagram by authors.

tangible representation of our vision for ODK 1.5 with the stakeholders. Figure 2 represents the mockup we created in Figma and the prototyping between screens.

Figma was selected for the UI mockup due to its widespread adoption in the industry, our existing familiarity with the platform, and its capabilities for comprehensive design and clear visualization of connections between screens. Presented in figure 3 are three examples of our screens: Home, Form Management, and Question.

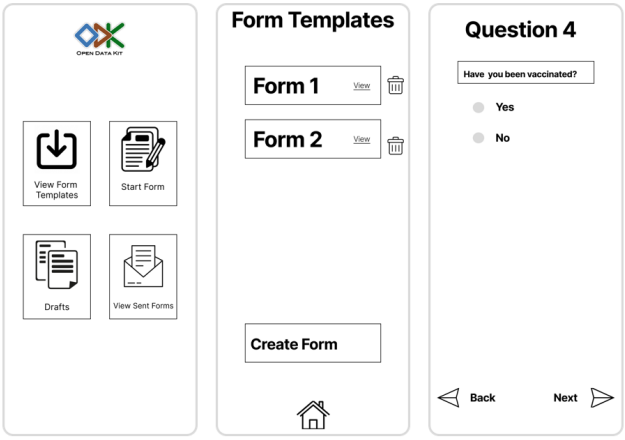

Figure 3: Example of three screens on the UI mockup. Diagram by authors.

4.2 Frameworks and Architecture

With a clear understanding of the current landscape and goals for design, we spent time researching frameworks that could support our goals for ODK in the modern world. Our ideal framework needed to be compatible with multiple mobile operating systems,

adaptable to the fast-paced environment of modern tech, maintainable, and supported by a strong community of developers. One of the struggles that we saw in modern ODK was the need for its developers to maintain compatibility with Android devices. One of our goals for the framework was the ability to maintain compatibility with modern services without requiring direct intervention from the developers. After compiling a comprehensive list of potential frameworks, we narrowed down our options based on the availability of libraries and tools. We wanted our framework to have a robust ecosystem that would allow us to build a versatile and efficient data collection tool within the time constraints of the quarter.

For our capstone solution, we chose Flutter as the framework for the reimplementation of ODK. Flutter is an open-source UI software development kit created by Google. Especially since Flutter is part of the Google ecosystem, we are hopeful that any updates to the Android OS will also have reflected changes in the Flutter library, maintaining compatibility without needing direct intervention from the developers. Flutter is object-oriented, a feature that allows for us to build manageable and extensible code. More importantly, Flutter is compatible with multiple mobile operating systems, aligning with our goal of supporting multiple operating systems. This compatibility ensures that ODK 1.5 can reach as many users as possible, regardless of the device they use. Flutter also provided many useful features beyond multi-system support. For example, a goal of ODK 1.5 is to upload the data collected onto the cloud, for data safety and access. Flutter supports complete integration with the Firebase, a set of backend cloud computing services developed by Google. We chose Firebase because it is relatively simple to integrate, especially because with our timeline speed is a priority as we test how quickly we can implement the majority of the capabilities of ODK. We chose Flutter as our framework as we are confident it has the capabilities to help us build a data collection application that is not only versatile but user-friendly and accessible to all.

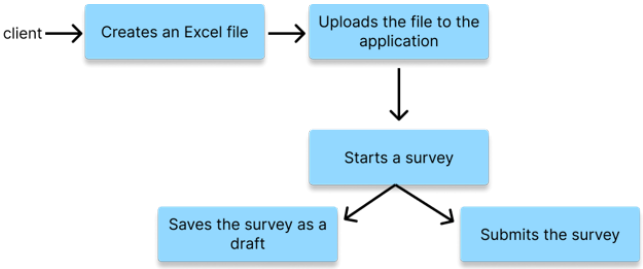

Figure 4: Model of user flow through ODK 1.5. Diagram by authors.

In addition to the UI mockup, we also created class diagrams, user flow diagrams, and data flow diagrams. The class diagrams helped us align on the system architecture and relationships between the different classes. We created user flow diagrams to expand on the user's journey through the application and data flow diagrams to better understand how the data would be processed and stored in the system. The user diagram in figure 4 to help illustrate the client path in the overall solution. The class diagrams and data flows are in the **Implementation Details** section following.

4.3 Initial Prototype and MVP

Our initial solution heavily depended on a third-party library within the Flutter environment, namely survey kit. Although this approach facilitated rapid iteration, enabling us to develop the initial prototype by the second week of development, it also resulted in limited flexibility. Presented in figure 5 is a subset of screens from this initial prototype.

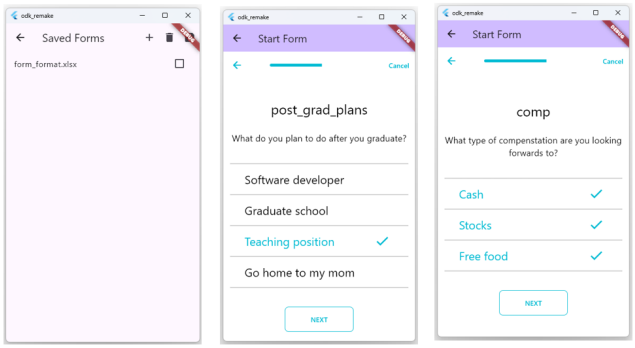

Figure 5: Three sample screens from the initial prototype. Image by authors.

In figure 5, it's challenging to distinguish between multi-choice and single-choice options. To tackle this issue, we made adjustments to a portion of the package. However, this fix wasn't sufficient. The package didn't support conditional logic or intermediate data storage, which became problematic for the more complex aspects of our product and data management requirements. Thus, we decided to remove our dependency on the package and rewrite the code to align with our original question architecture. This move allowed us to regain control over our system's functionality and better adapt it to our evolving project needs and technical specs. In figure 6 is a subset of screens from the complete minimum viable product.

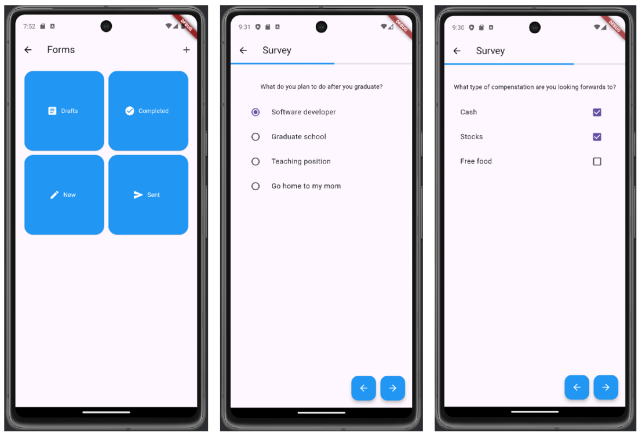

Figure 6: Three sample screens from the minimum viable product and final prototype. Image by authors.

The reimplementation of ODK in the modern world involved not only a careful selection of the technology stack but also a

meticulous design process. By leveraging modern design tools and methodologies, we were able to create a robust, efficient, and user-friendly data collection tool that is ready to meet the demands of the modern world.

5 IMPLEMENTATION DETAILS

The implementation of ODK 1.5 involved several key steps. We split up the initial development work so that the work could be completed concurrently: retrieving file, parsing data, implementing survey logic. Then, we worked on data flows, navigation, integration, and accessibility features.

5.1 Retrieving File

The first component was to implement the retrieval of the Excel file that contained the survey data. The application is retrievable from a web source or a local directory. We created different statuses for an Excel file: new, draft, completed, sent. The 'new' status represents a copy of an empty Excel that has been uploaded. Uploaded Excel files automatically have a "new" status. The "draft" status represents a copy of a partially-completed Excel file that has been declared as a draft by the user. An Excel file has the "draft" when the user clicks "Save as a draft" while completing a survey. The draft is cached locally in the application and not saved to the user's directory until the user marks the survey response as completed. The "completed" status represents a copy of a fully-completed Excel file that has been declared as completed by the user. An Excel file with the "completed" status will be automatically saved to the user's local directory. The "sent" status represents a copy of the Excel file that has been completed and also sent to the Firebase storage. To retrieve the Excel from the local directory, we used the filepicker library from Flutter to open a dialog to the file explorer of the user's local system. The name and the location of the file are saved to the application's cache, and the file is saved with the status "new".

5.2 Parsing File

The second component was to implement a parser for the Excel file. For the capstone, we chose the XLSForm format for the Excel file. The XLSForm was adopted and integrated into ODK by the ODK community, when the general population pivoted towards using Microsoft Excel in research for data collection [10]. One of our goals in ODK 1.5 is to bypass the conversion to XML from Excel and directly parse the Excel form. ODK is unable to directly parse Excel files, instead ODK converts the Excel form into ODK's XForm, an XML format. The conversion to XML format is a byproduct of the evolution of technical literacy. With Flutter, we can directly retrieve data from Excel, and the intermediate conversion to XForm becomes unnecessary.

The XLSForm gave its users the ability to create surveys, where each row represents a question in the survey. The XLSForm format supports over 25 different question types, including text, multi-choice, GPS, and more. An example is shown in figure 7. The example does not show all possible options but serves as a snapshot example. For the capstone, we constrained the type of question types supported to: integer, decimal, text, multiple-choice questions where only one answer can be selected, multiple-choice questions where multiple answers can be selected, date, and time. The parser

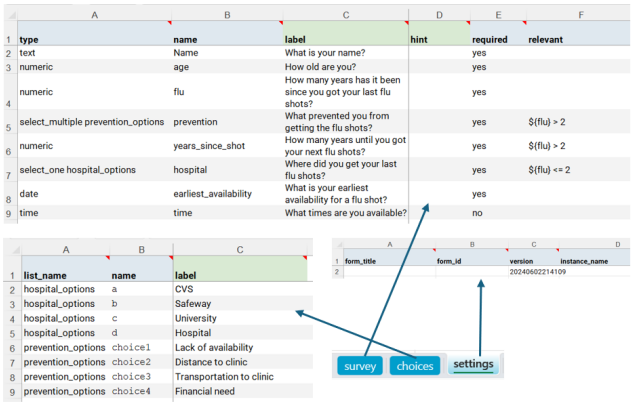

Figure 7: Example XLSForm about flu shot data. Diagram by authors.

iterates through every row in the Excel file and creates a new question for each row. It identifies the type of question, the question's unique identifier, the question itself, and other parameters of the question, such as whether the question is required and whether the question should be shown based on answers to prior questions. In the situation where the question is a multiple-choice question, the parser must find the answer choices corresponding to the question in the different sheets in the Excel file. Then, the parser reads and stores the answer choices associated with the question. The Excel files are parsed into lists of questions once the user opens an Excel file as a survey.

5.3 Implementing Survey Logic

The third component was to implement the survey logic itself, from implementing the survey questions to rendering the UI of the questions. We designed a class that encapsulates the question metadata and an UI widget. The widget is a dynamic component of the class, changing its form based on the type of question it represents. This design allows for a wide variety of question types, each with its own unique widget, all under the umbrella of a single class. The type of widget to associate with the question is derived from the question type in the question metadata. The question metadata is directly retrieved after parsing the Excel file. This approach, combining OOP with dynamic widgets, provides a robust and adaptable framework for handling diverse question types and metadata in our system. This design also allowed us to quickly extend and maintain similar behavior among the different question types. For example, if we were to add an option to upload an image to the survey, we would only need to add an additional question type and a custom UI widget to display the front-end for how the user would upload an image. All the other logic, such as parsing the data or determining whether the question should be rendered based on prior answers, remains the same. The architecture of the question widgets is in figure 8.

The survey is designed as a stack where the top-most value is the most recently answered question. The survey iterates through the list of questions, rendering the next/previous question when the user clicks on the next/previous arrow. The responses to the

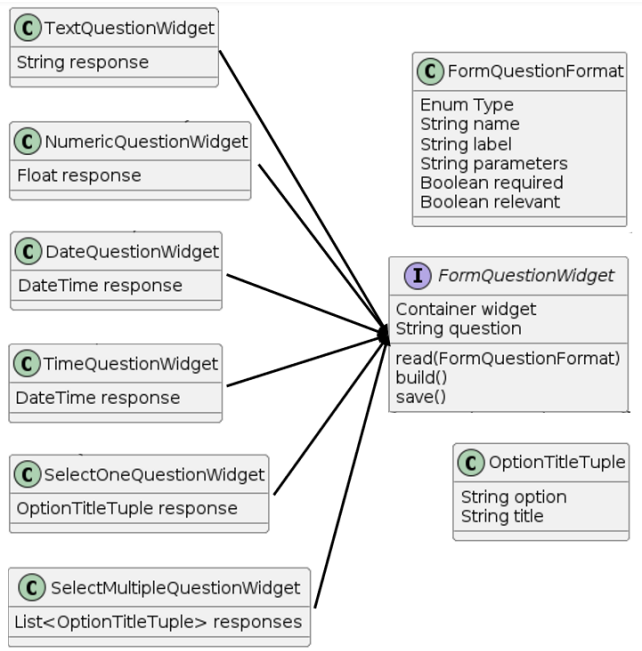

Figure 8: Model of data flow through ODK 1.5. Diagram by authors.

questions are cached locally in the application when the user clicks on the next/previous arrow.

An important feature of the survey is to show different questions based on the user's responses, allowing for a more customizable and personal experience. The branching logic in XLSForm depends on the answers of the prior questions. For the capstone, we limit the type of branching logic to only support relational operators (">", "<", "<=", "=>", and "=") for numerical values and the equality operator ("=") for string values. We parse the boolean expression in the question metadata into three parts: the operand on the left-hand side, the operator, and the operand on the right-hand side. From the operand on the left-hand side, we parse out the question whose answer determines whether the current question should be rendered. If the boolean expression evaluates to true, then the question is rendered. Otherwise, the question is not rendered, and the logic for the survey continues to iterate through the list of questions until the next suitable question is found.

5.4 Saving Data

The final feature of the project is the ability to save the data. We chose Firebase as our cloud-based database. The main reason for this choice is because of the ease of implementation. However, Firebase is a strong choice for a database, if we were to continue building ODK 1.5. Firebase supports automatic scaling, which is highly necessary for the sheer volume of users and data volume of ODK, robust security features, integration with Google, and offline support. In our implementation, once the user submits the survey responses, we save the responses to the local file system first. Once the user has an internet connection, then they are given the option to "send" the file to the cloud. When the user chooses to send a file to the cloud, then we use the Firebase SDK to upload the file to a pre-specified Firebase database. We monitor the progress of the upload in the background. If the upload fails, then we notify the user and wait for their choice on whether the upload should be retried. As part of the integration process, we diagrammed the data flow in figure 9. This helped connect the components we were building and ensure data reliability and integrity.

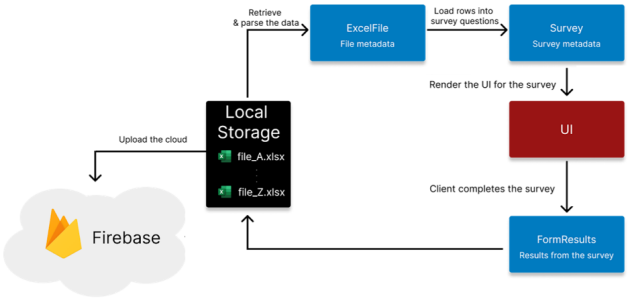

Figure 9: Model of data flow through ODK 1.5. Diagram by authors.

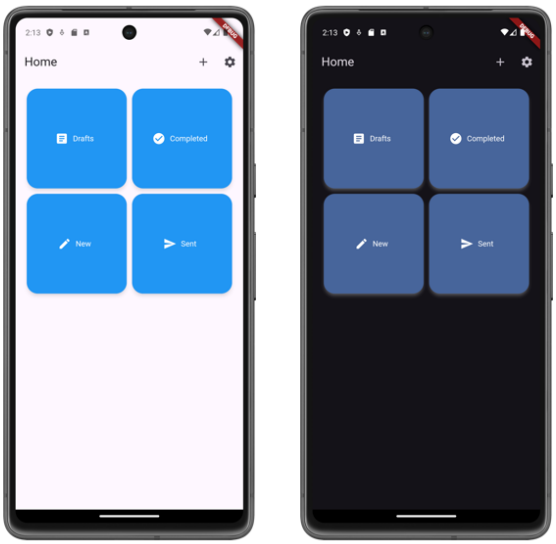

Figure 10: Screenshot of home page in light mode and dark mode. Image by authors.

5.5 Accessibility

Accessibility and additional features were our next priority after developing the basic product. Ensuring an accessible and intuitive UI was a key aspect of our re-implementation. The app now supports both Android and iOS text-to-speech functionalities, enhancing usability for visually impaired users. Additionally, the interface resizes automatically based on zoom and screen size, providing a

seamless experience across different devices and for various users' viewing needs. These enhancements reflect our commitment to making ODK more inclusive and user-friendly. As part of this, we also implemented themes to toggle between a light and dark mode. In figure 10, there is a screenshot of the home page in light mode and dark mode.

6 DISCUSSION AND EVALUATION

The project capstone saw several successes. We chose Flutter because of its extensive support and developer community, building 3rd party libraries. By the second week of development, we had achieved significant milestones: loading an Excel file from local storage, parsing the Excel file into a list of questions, and rendering a simple survey. We were able to quickly identify areas in the survey that could potentially cause problems, allowing us to address these issues early in the development process.

There were, however, some challenges. One issue was with LaTeX formatting on documents. While LaTeX is a powerful tool for typesetting, it can be complex and time-consuming to get the formatting just right. This was particularly true in our case, where we needed to create complex documents with various formatting requirements. Another challenge was the lack of flexibility offered by some packages.

The ODK project has been a journey of seeking feedback and refining our development processes. Throughout this project, we have consistently engaged with stakeholders to gather invaluable insights, which have guided our iterative approach to development. Key learnings include the importance of exploring new technologies to enhance our capabilities and the necessity of being adaptable in the face of evolving requirements and unexpected challenges. This project has also underscored the need to embrace ambiguity, as it often leads to innovative solutions and improvements. By iterating on feedback and remaining open to new possibilities, we have been able to make significant strides in modernizing our data collection tools and methodologies. Overall, despite some challenges, we were able to develop a minimum viable product that works decently well. The project served as a good base to model off of, and the experience provided valuable insights into the development process. Moving forward, we aim to build on these learnings to improve future iterations of the project.

6.1 Flutter Packages

Having libraries at hand was very beneficial, as it enabled us to rapidly develop a first iteration of the project. We went through several iterations of the project. Our first several iterations revolved around how we were parsing the data from the Excel file, and, after we created a modular solution, we were able to quickly add logic to parse four additional types of survey question formats. Several of our iterations were on the UI of the survey. When we first used the survey package built by a 3rd party, the UI did not match up nicely with our mockup. The single-choice and multiple-choice question had the same look, without anything indicating that only one option or multiple options were allowed. We quickly fixed this by modifying the single-choice question to use radio buttons, a common indication of only one answer option. We also modified the multiple-choice question to display check-boxes, a common indication of having multiple answer options. Overall, the development process was very smooth due to taking advantage of Flutter's libraries early on to quickly iterate and find problematic areas early on.

While these packages are useful, they often come with constraints that limit customization. One of the biggest limitations of the package was its inability to access intermediate results. Without being able to access the intermediate results, we were unable to implement the branching logic based on the answers from prior questions. This led to a situation where we ended up writing the logic and structure for the survey ourselves. We wrote logic for saving the data, moving to next and previous screens, branching logic based on answers from prior questions, and mechanisms for rendering custom UI based on the question type. Another challenge was the ambiguity of the capstone project. We only had a general idea of what needed to be done, and we explored the domain of the project ourselves, figuring out new restrictions and solutions as we progressed. The ambiguity of the project also allowed us to be creative with our solution. When we ran into the inflexibility of the packages, rather than working around the restrictions of the package for the survey, we wrote our own object-oriented solution to create a survey.

6.2 Discussions with Developers

Our engagement with the original developers provided invaluable insights into the genesis and motivations behind ODK's development. These dialogues were instrumental in illuminating the roots of ODK and the driving forces behind its creation. Moreover, through these discussions and collaborative efforts, our team identified key areas for potential expansion and refinement within our project and beyond.

Our discussion on May 7 with Waylon Brunette centered on the objectives and challenges faced in implementing ODK Collect. While recognizing the presence of multimedia features in the tool, there were acknowledged limitations, particularly concerning its support. Deliberation ensued over file formats, highlighting the shift from XLS to XLSX post-2007 and the adoption of ODK X. The conversation also touched upon the significance of X Form as a W3C standard and the current reliance on the JavaRosa library, with a suggestion to explore alternatives. The original vision emphasized moving away from database dependency, citing concerns about complexity and accessibility. Instead, there was a call to streamline processes, enhance data quality, and ensure security. The discourse expanded to include considerations like GDPR compliance and the evolving landscape of data representation. Questions arose about storage solutions, especially in regions with limited cloud infrastructure like Uganda. Ideas such as caching and Firebase emerged, along with the necessity for seamless syncing across devices. Overall, the discussion underscored the need for innovative approaches to address both technical and regulatory challenges in data collection and management.

The discussion on May 23rd, led by Carl Hartung, delved into the concept of modernization within the context of data collection technologies, particularly focusing on ODK. Key considerations revolved around streamlining processes to simplify implementation and leveraging contemporary technology like Firebase for

efficient development. The conversation highlighted the importance of swiftly incorporating a wide range of capabilities into the system and pondered the next steps in the project's evolution. Integrating with various databases was deemed feasible, with an emphasis on exporting data to Excel seamlessly. There was a historical perspective provided regarding the evolution of the tool, with insights into user-driven modifications such as the introduction of raw Excel export to accommodate form formatting preferences. Although the necessity of the X Form standard was debated, its role as a helpful industry standard was acknowledged. The discourse extended to exploring strategies for transitioning from paper-based to digital data collection, addressing challenges like data loss and ensuring cross-platform compatibility. Reference was made to related projects like CommCare, shedding light on integration issues and future aspirations for bi-directional communication and enhanced data visualization capabilities. The discussion also touched upon the need for modularization to replace existing packages, facilitating smoother interactions with data visualization tools, and ensuring data usability through validation and verification processes. Furthermore, the potential for gamification in data collection and the relational dynamics between questionnaires were contemplated, emphasizing a broader objective of building a more generalizable data collection ecosystem beyond the confines of ODK.

7 FUTURE WORK

The culmination of our capstone project marks the achievement of a significant milestone as we finalize the development of our application. As we reflect on this accomplishment, we recognize the importance of real-world testing to validate our product's effectiveness and usability. Given more time, our priority would be to engage with actual researchers, providing them with hands-on access to our application within their working environments. This direct engagement would enable us to gather invaluable insights, identify any usability issues, and refine our features based on user feedback. Moreover, our commitment to inclusivity drives our desire to expand accessibility features within our application. One key aspect we aim to address is the integration of translation capabilities, allowing users from diverse linguistic backgrounds to fully utilize our platform. By enhancing accessibility, we strive to ensure that our application can be seamlessly utilized by a broader audience, fostering inclusivity and usability across various demographics.

While we are proud of the progress we have made with our minimum viable product, we also acknowledge that there is room for further refinement and enhancement. With additional time, resources, and thorough testing, we envision implementing improvements and optimizations to elevate the functionality, user experience, and overall quality of our application. Despite the completion of this phase, we remain committed to continuous iteration and enhancement, driven by our dedication to delivering a valuable and impactful solution. As we explored this space, we found many different areas to explore and build in. The following points are a few of the areas where we believe have the potential to be explored and the considerations for future iterations. These different areas of exploration represent the roadmap for our application's evolution, with a focus on enhancing user experience, data quality, and overall functionality.

7.1 Additional Features

One additional feature to build onto ODK would be bi-directional communication between the user completing the survey and the user managing the surveys. Enhancing user engagement through bi-directional communication will allow for a more interactive experience. This will enable users to not only submit data but also receive feedback, creating a dynamic flow of information. It would also enable multiple roles: enumerators and district managers.

As we have seen before, many different mobile operating systems are being used in research, and our reimplemented version of ODK supports multiple systems. Another potential expansion along this avenue would be the revival of a web-based option. A web-based option will cater to users who prefer or require desktop access, increasing the application's versatility.

Another interesting area of exploration came up when we were talking to Dr. Hartung. Introducing gamification elements into data collection could incentivize user participation and improve the quantity of data gathered. However, there was a concern that users would try to "game" the system in order to win more awards. A potential area of research would be to figure out how to create robust validation methods to help identify outliers and anomalies, ensuring data integrity.

7.2 Possible Integrations

Outside of how the application could be improved, we also considered hardware modifications to the data collection process. Future versions could explore the integration of sensors and the collection of more complex data types, such as health indicators. This could provide more diverse data points in the dataset and give deeper insights.

One area that is lacking in digital data collection is how restricted the data is presented. For example, with paper data records, the surveyor could add notes within the margins where the response is relevant to the research but does not fit within one of the questions. For future iterations of ODK, more flexible data inputs can be explored. With more flexible data inputs, there would also need to be a well-established methodology with existing data visualization tools. The methodology should ensure that data is also presented in an insightful manner.

We also considered exploring the datasets themselves, and we propose two changes to the dataset's inner-relationships and data gaps. Oftentimes, there are relationships between different surveys. Exploring the relational behavior between different questionnaires may uncover patterns and correlations, enhancing the data's richness. We also considered the feasibility of pulling data from web sources to enrich our datasets. This could help fill any gaps in the data and provide a better overview of any relationships presented by the data.

Finally, one common theme we saw with prior versions of ODK was that it was a standalone service. Its framework was not supported by a large tech ecosystem that actively supported backwards compatibility. Researchers with different needs had to contact the developers to identify whether it was possible to program a feature for their research use case. Many features in ODK, such as inputting data through Excel, were added once the need arises. Moreover, it takes many man-hours, possibly more hours than the

researchers have allocated, to implement, test, and deploy a new feature built on ODK. A potential area of exploration would be to create a more generalizable version of ODK that could adapt to various data collection needs, streamline the research process, and improve the researchers' experience with ODK. Transitioning from siloed operations to a modular framework could help with building new features faster and enhance the system's adaptability.

8 CONCLUSION

We started the remaking ODK project from scratch. This allowed us to incorporate modern design principles and leverage the newest technologies. We were able to successfully meet our goals for the capstone project. We set out with the goal of determining how quickly we could implement the basic functionalities of ODK: taking in a survey format in Excel, creating a survey, saving the data to the local storage, and sending the data to a remote server when requested by the user. Our journey for the past 10 weeks was successful. The project not only showed that a re-implementation is feasible, but it can also offer significant benefits. For example, we were able to bypass the process of converting Excel files to an XML format because we were not limited to the input format, like ODK. The experience was also quite informative as an experience representing the culmination of our careers in the Computer Science bachelor program. We were able to take advantage of tools we learned how to use in other courses and further develop technical and non-technical skills.

The success of this project suggests that other organizations facing similar challenges with limitations from old or out-dated technology could also benefit from this approach. It shows the potential of re-implementation as a viable strategy for software revitalization, paving the way for future research and practical applications in this area.

ACKNOWLEDGMENTS

To Richard Anderson and Lisa Orii, for the 1:1s, advice, and arranging guest speakers for our class. Thank you to Waylon Brunette and Carl Hartung for reviewing our proposal for ODK Collect 1.5 and providing useful context for the development of and need for ODK.

REFERENCES

[1] Zulfiqar Ali and S Bala Bhaskar. 2016. Basic statistical tools in research and data analysis. *PubMed Central* 60, 9 (Sept. 2016), 662–669. https://doi.org/10.4103/0019-5049.1906232

[2] Richard Anderson. 2024. Course Introduction: ICTD Capstone Software Design for Underserved Populations. Retrieved June 4, 2024 from https://courses.cs.washington.edu/courses/cse482b/24sp/lectures/Lecture01/Lecture01.pdf

[3] Ronak Jain, Himansi Bhatt, Nithya Jeevanand, and Praveen Kumar. 2018. Mapping, Visualization, and Digitization of the Geo-Referenced Information: A case study on Road Network Development in Near Real Time. *International Research Journal of Engineering and Technology (IRJET)* 5, 9 (Sept. 2018), 845–850. https://www.irjet.net/archives/V5/i9/IRJET-V5I9153.pdf

[4] Andreas Kipf, Waylon Brunette, Jordan Kellerstrass, Matthew Podolsky, Javier Rosa, Mitchell Sundt, Daniel Wilson, Gaetano Borriello, Eric Brewer, and Evan Thomas. 2016. proposed integrated data collection, analysis and sharing platform for impact evaluation. *Development Engineering* 1 (Jan. 2016), 36–44. https://doi.org/10.1016/j.deveng.2015.12.002

[5] Temina Madon, Ashok J. Gadgil, Richard Anderson, Lorenzo Casaburi, Kenneth Lee, and Arman Rezaee (Eds.). 2023. *Introduction to Development Engineering*. Springer Cham, Chapter Chapter 23: The Open Data Kit Project, 613–637. https://doi.org/10.1007/978-3-030-86065-3

[6] ODK. 2023. ODK for Clinical Research. Retrieved June 5, 2024 from https://getodk.org/success-stories/clinical-research/

[7] ODK. 2023. ODK for Field Mapping. Retrieved June 5, 2024 from https://getodk.org/success-stories/field-mapping/

[8] Kristin Osborne. 2018. Allen School recognizes Yaw Anokwa and Eileen Bjorkman with Alumni Impact Awards. Retrieved June 4, 2024 from https://news.cs.washington.edu/2018/06/08/allen-school-recognizes-yaw-anokwa-and-eileen-bjorkman-with-alumni-impact-awards/

[9] Nissrine Souissi Patrick Loola Bokonda, Khadija Ouazzani-Touhami. 2019. Open Data Kit: Mobile Data Collection Framework For Developing Countries. Published By: Blue Eyes Intelligence Engineering and Sciences Publication. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)* 8, 12 (Oct. 2019). https://doi.org/10.35940/ijitee.L3583.1081219

[10] XLSForm.org. [n. d.]. History. Retrieved June 4, 2024 from https://xlsform.org/en/#history