Software and System Design II
Final Report

Development of an Application for Secure Digitization of Consultation Forms

Chandler Klein and Garrett Ruffner

WESTERN MICHIGAN UNIVERSITY

Computer Science
Western Michigan University
Kalamazoo, MI
08 December 2020

Software and System Design II Final Report

Development of an Application for Secure Digitization of Consultation Forms

Chandler Klein and Garrett Ruffner

Abstract

Paper consultation forms in healthcare are often cumbersome and not easily digitized, even with the help of staff members. An Android application for tablet devices was created using an Integrated Development Environment (IDE) called Android Studio and MyScript. The application allows medical staff to securely connect to a local file server, pick a customizable template to generate a patient consultation form, fill out a form using handwriting recognition, and export a completed form. The application is Health Insurance Portability and Accountability Act (HIPAA) compliant and performs all operations securely to ensure patient privacy. The completed application provides a solution for efficient and secure digitization of patient forms and improves the quality and volume of local neuropsychology consultations.

Contents

List of Figures **3**

1 Problem Statement **4**
1.1 Need . . . 4
1.2 Objective(s) . . . 4
1.3 Terms, Acronyms, Glossary . . . 6

2 Problem Analysis and Research **7**

3 Requirements **9**

4 Standards and Constraints **11**
4.1 Applicable Standards . . . 11
4.2 Constraints . . . 11

5 System Design **13**

6 Testing **16**
6.1 Usability Testing . . . 16
6.2 Accessibility Testing . . . 16
6.3 White-Box and Black-Box Testing . . . 16
6.4 Conformance Testing . . . 17

7 Results **18**
7.1 Realization of Requirements . . . 18
7.2 Realization of Standards and Constraints . . . 19
7.3 Testing Results . . . 19

8 Future Work **20**

9 Conclusion **22**

10 References **24**

A Development Costs **25**
A.1 Institution Costs . . . 25
A.2 Sponsor Costs . . . 25
A.3 Team Costs . . . 25

List of Figures

5.1 An overview of the project's system design 13
5.2 Server file structure 14
5.3 Tablet-server interactions 15

Chapter 1

Problem Statement

1.1 Need

Using paper forms to record patients' medical data is a simple, albeit cumbersome solution to an ever-present problem. One advantage of using paper is that storage is relatively secure, since physical access is required to retrieve any information. Unfortunately, file cabinets take up a significant amount of physical space. It is often necessary for medical documents to be shared between healthcare professionals. Sharing paper documents is time-consuming because documents must be individually scanned and sent digitally.

A more modern approach to the issues of patient information storage is the usage of an Electronic Health Record (EHR) system. These systems allow for storage scalability far beyond the capabilities of file cabinets. Another advantage of EHR systems is the ease of use they provide to access the data they contain. This level of access also allows healthcare providers to share documents effortlessly. One concern with EHR systems is digital security. These concerns are addressed by the Health Insurance Portability and Accountability Act (HIPAA) regulations that are placed on systems containing healthcare information.

Unfortunately, EHR systems can cost millions of dollars for large healthcare organizations. Even for smaller organizations it is common to see costs ranging in the thousands of dollars, with significant recurring monthly bills. Costs of this magnitude create a vast barrier of entry for small healthcare practices. This barrier of entry often forces medical offices to continue using paper medical records while large organizations improve the speed and quality of service they provide using expensive EHR systems.

1.2 Objective(s)

The main purpose in developing this application was to create a secure, robust, inexpensive system for digitization of paper medical forms. The first and most important part of this objective was to create a safe, secure system. In order to be accepted in any medical context relating to patient information, applications of all kinds must adhere to HIPAA rules and guidelines.

Besides security, the application needs to consistently perform operations previously in the client's workflow in a more efficient manner. For example, to improve the efficiency of transcribing a paper document into an electronic one, the need for

paper documents was completely removed from the equation. Paper was replaced by a tablet, and the role of the transcriptionist was reduced to simple error-checking of a digital document. This massive improvement in workflow demonstrates a central focus on efficiency that was present since the beginning of application development.

Lastly, the application had to be inexpensive. EHR systems, which are at times extravagantly expensive, were out of the question for a small practice like our client's. Because of this, a main focus during application research and analysis was to determine the best ways to keep costs low. Additionally, performant development needed to be done such that costs were not kept low at the expense of quality. A main goal during application development was to achieve high quality and usability not unsimilar to what EHR systems provide.

1.3 Terms, Acronyms, Glossary

Acronyms

API Application Programming Interface. 8, 18, 22

EHR Electronic Health Record. 4, 5, 7, 9, 11, 22

HIPAA Health Insurance Portability and Accountability Act. 1, 4, 8, 9, 12, 17, 18, 22

IDE Integrated Development Environment. 1

JSON JavaScript Object Notation. 14, 18, 20

SFTP Secure File Transfer Protocol. 8, 11, 12, 14, 17, 19, 23

Glossary

Android An open-source operating system used for smartphones and tablet computers. 1, 7, 8, 12, 17, 19

Android Studio The official integrated development environment for Google's Android operating system. 1, 11

BuildFormer A software library for building forms dynamically in Android written by Girish Raman. 7, 12, 19, 20, 23

docx4j docx4j is an open source (ASLv2) Java library for creating and manipulating Microsoft Open XML (Word docx, Powerpoint pptx, and Excel xlsx) files. 12, 18

Java A class-based, object-oriented programming language. 11

JSch Java Secure Channel—a library allowing SFTP connection to a remote machine. 8

MyScript A popular handwriting recognition engine deployed in mobile devices, cars, interactive whiteboards, etc. 1, 8, 11, 12, 20–22

SSH Secure Shell is a cryptographic network protocol for operating network services securely over a network. 11, 12, 17

vigintillion A number equal to 10^{63}. 22

Chapter 2

Problem Analysis and Research

The main problems that needed to be solved (each with their own subproblems) included form generation, handwriting recognition, patient information security, and server connection. Each of these problems were quite unique, needed to be solved digitally, and required plenty of research to understand how to approach them.

The central problem to be addressed and the main function of the application was to be able to fill out consultation forms digitally. There was a big caveat though—the final product needed to keep the interaction between the medical provider and patient as personal as possible. An issue that our client has with EHR systems is that they generally require the use of laptops or "heavy-duty" workstations in order to run the software they provide.

For instance, if you've ever had an annual physical, your healthcare provider may have faced their computer to enter your information, then turned to face you, then turned away to enter more of your information, and so on. Our client firmly believes that these kinds of interactions are impersonal, fragment attention, and obscure the human connection between medical provider and patient. This is a big reason (besides the cost of EHR systems) why our client still relied on paper consultation forms before completion of the application.

Knowing all this, research was done to find the best way to record data digitally while preserving the human connection and attention that our client expected to give during patients' consultations. After ruling out options like laptops and other bulky devices, we decided on a Samsung tablet that is similar in profile (except in thickness) to a normal piece of paper and allows for handwriting on its screen with an "S Pen". Another crucial reason we chose a Samsung tablet is because it runs on Android, providing us with a developer-friendly avenue to create an application with a programming language we were comfortable with, in contrast to the more restrictive Apple ecosystem.

Once the issue of hardware was out of the way, research was done on methods of generating forms. It was determined that generating a consultation form based on an input file that specified the type of form element (checkbox, input, radio buttons, etc.) and a description, among other options, would give our client the most freedom in creating whatever digital form they wanted. Using this format, our client can easily create, update, and delete any input template files they have created. This method also eliminates the need for a persistent internet connection, as a template file can simply be saved to the device's internal storage. After researching ways to parse template input files we discovered a library called BuildFormer, which enabled

us to display a user interface that mirrored the contents of the consultation form template file.

Another problem that was central to the usage of a tablet and pen was the possibility of handwriting recognition in our application. We soon discovered that handwriting recognition on a low level is a relatively costly and complicated operation and that development of a custom solution would be extremely difficult. After looking into both open-source and third-party options, we decided that MyScript would be our best option. MyScript is unfortunately (but unsurprisingly) not a free service, but it provides a high-quality, private solution for the issue of handwriting recognition in a HIPAA compliant application.

Perhaps the most important consideration in the development of the application was the handling of sensitive patient data. Research was done on the HIPAA to ensure that all necessary guidelines and rules were being followed in order to protect patient privacy. In order to be HIPAA compliant, the application has to have administrative, physical, and technical safeguards. Most safeguards were either already in place at the client's office or were implemented by default hardware configuration—including automatic lockout after a certain duration, facial recognition, password, etc.

Since the application can be used to fill consultation forms without a persistent network connection, the issue of handling and storage of patient data presented itself. In order to adhere to the HIPAA, any patient form data that is stored through the application is encrypted using Android encryption Application Programming Interfaces (APIs). The only way for consultation form data to be read in a human format is to log in to the tablet and transfer the files to a specified local file server using a Secure File Transfer Protocol (SFTP) connection. Upon completion of a patient consultation form, if a valid network connection is possible, the application will instead immediately transfer the files using a SFTP connection and not store any sensitive data. These privacy safeguards were found to be the most effective methods for storage and transferring of sensitive data on a mobile device.

Usage of a file server to send and receive files to and from a tablet was the concept least familiar to the team in terms of implementation. The first problem in this phase was getting an input template file corresponding to a consultation form onto the tablet in order to create a matching user interface. We researched available options and found JSch. Using JSch, we were able to implement secure transfer of input template files to the tablet from the server, and secure transfer of completed consultation form documents to the file server.

Other problems that were researched and solved include saving of user preferences, listing of files on the tablet and server, general user interface navigation in Android, public key QR code scanning using the camera, virtual machine setup with a script, file structure conventions on the tablet and server, and file format of input and output documents.

Chapter 3

Requirements

Enabling superior provider–patient openness while maintaining digitization capabilities was an important requirement of this application. In neuropsychological consultations it is necessary to closely observe and communicate openly with patients in order to best assess their cognitive function and behaviors. A consultation performed properly yields critical information about the functional integrity of the patient's brain. Our client believes that an adequate level of communication between a provider and patient is not achievable with an EHR keyboard-based solution since continuously turning or looking away from a patient to type on a keyboard is impersonal and fragments attention. This important requirement was the first to be addressed and guided most of the hardware and technical specification decisions.

Security is a top priority any time patient data is being stored or transferred. A main requirement of the project was to follow relevant Health Insurance Portability and Accountability Act rules and guidelines since the application would be used in a medical context. Learning about and implementing these rules and guidelines required research on secure data storage, transfer solutions, as well as administrative, physical, and technical safeguards. This requirement was the most important—if these standards were not followed, it wouldn't matter how great the application was, it would never be accepted in any medical environment due to the risks to patients' privacy.

Another critical requirement was to seamlessly integrate the application into the client's administrative office. We discussed with the client how they develop and share consultation forms in their office and with other medical service providers. Our application was designed to eliminate the cumbersome and time-consuming parts of their workflow that surround paper-based forms—like manual transcription of text to a digital format, for example.

Our client's consultation form is frequently adjusted to improve readability and better address patients' needs. Because of this, a requirement of the application was that the configuration files that specify form elements and their contents needed to be easily created and updated. Our client's technical staff is not always immediately available, so this process needed to be achievable by someone with minimal technical ability.

The documents outputted by the application must be easily readable for any healthcare professional. Our client also needs the ability to share their professional assessment of the patient via a secure fax line. Our decision to have a user-defined consultation template file that form value responses are inserted into reflects the

need for a readable document that can be altered stylistically and still contain all necessary form data. The output document from the application is in a DOCX format that can easily be proofread and finalized by office staff.

Chapter 4

Standards and Constraints

4.1 Applicable Standards

The completed project requires the usage of a file server with an SFTP and SSH implementation to serve configuration files and receive completed consultation form documents. The file server must have folders to contain template input files and output documents. This server should of course be securely implemented using SFTP in order to protect the privacy of patients' documents. The project application file corresponds to development in Android Studio version 4.0.1 with Java version 8. Both USB and Bluetooth connections present security vulnerabilities for the application—standards for each are not addressed or implemented. A wireless connection to the same network as the file server is required. Additionally, a valid certificate must be present in the application in order to verify the user's MyScript account, which is used to process all handwriting recognition requests. The final production build of the application runs on our client's Samsung Galaxy Tab S6 tablet (model no. SM-T860) with Android version 10.

4.2 Constraints

Our application is designed to eliminate the need to fill out consultation forms on paper. The transition away from paper reduces the workload on some of the staff because transcription of the completed consultation form into a digital format is no longer necessary. The office that our application is integrating into is already equipped with a secure file server that can be accessed by the staff from any of their workstations. This allowed us as developers to implement the server as secure storage for completed consultation forms. Since a secure file server existed for long-term storage of documents, the security constraints for our project were limited to only two categories—file transfers to the secure server, and temporary storage of files in the tablet's internal memory. Another constraint for our application is the cost for maintenance—an initial requirement has been to keep this cost as low as possible in order to compete with traditional EHR systems.

To address file transfer security, our application uses an industry standard transfer protocol called SFTP. The SFTP protocol uses an SSH connection to transfer files. Using these protocols, we can ensure that both the authentication data and file contents are encrypted using AES-256 as the encryption algorithm. The only vulnerability present with this system is the unauthorized access of user credentials.

Two different methods can be used to establish an SFTP connection. One method implements a username and password combination, and the other an SSH key. The HIPAA only requires that one of these methods be used, but our application uses both as an added layer of security. Another potential issue is the storage of user credentials in the application. Entering in user credentials for every file transfer when using an SFTP connection would quickly become time-consuming—instead, user credentials are saved in an encrypted file on the tablet. This means that even when the application is closed or the tablet is turned off, the credentials will still be saved until the app is deleted. This was implemented using the built-in Android security class called "EncryptedSharedPreferences", which encrypts values using the AES-256 GCM algorithm.

To address temporary file storage security we used the class EncryptedFile from androidx.security.crypto.EncryptedFile. This class provides an easy way to create and read from encrypted files. This class is included in the same package as EncryptedSharedPreferences, so we were able to add file encryption to our application without introducing new dependencies. File encryption occurs seamlessly in the background of the application and is invisible to the user. When files are transferred to the secure file server they are checked for integrity then deleted from the tablet's internal storage.

In order to keep costs low, we focused on using open source projects as much as possible to develop the features that our client required. This is reflected by our use of the docx4j library to interact with Word documents, SFTP for secure file transfer, and BuildFormer to dynamically generate interactive forms. The only premium software used in this project is MyScript, which is used to convert handwriting into text values. MyScript has very reasonable pricing models that are based on the number of requested handwriting-to-text conversions per month. The conversions are done locally on the physical device, ensuring that patient information is not sent elsewhere for processing.

Chapter 5

System Design

A high-level design overview of the application is given in Figure 5.1. A patient consultation is represented by all of the elements in the blue box. In the center of the box is the tablet used to enter all patient consultation data. Any completed consultation form documents that are not immediately transferred to the secure file server are stored as encrypted, human-unreadable files in the internal storage of the device.

An SFTP connection, depicted at the bottom right-hand corner of the blue box, is used to transfer consultation form data to the secure server and to receive consultation form configuration files on the tablet. These configuration files are used to display the necessary questions and responses of the consultation form on the user interface and to insert the responses of the form into a formatted DOCX file.

The green box represents the administrative office of the client's clinic, which houses the secure server and any other devices like PCs and fax machines. In order to send completed consultation forms to other medical providers, administrative staff have protected access to the secure file server. Once they've retrieved the desired consultation information, they can proofread and finalize it before it's sent off.

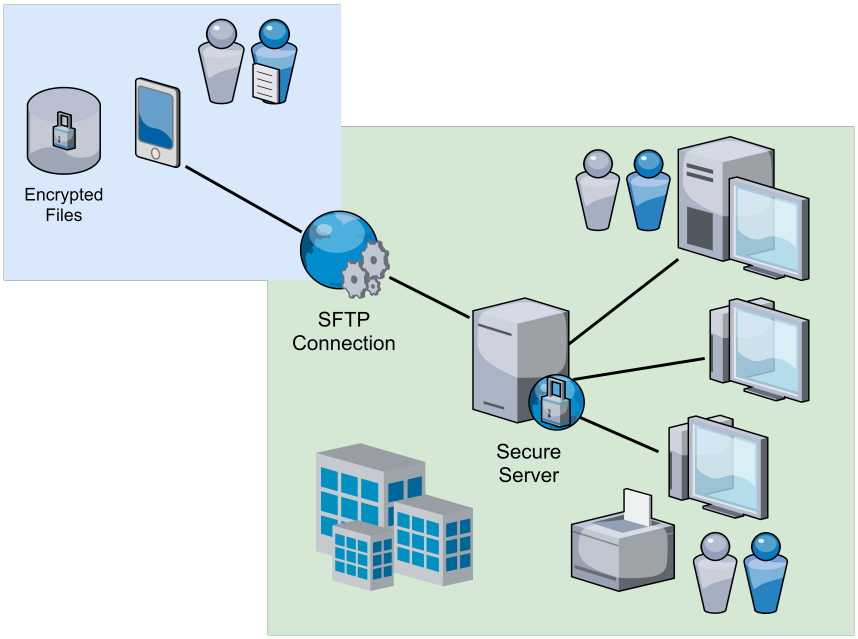

Figure 5.1: An overview of the project's system design

Figure 5.2 displays a general overview of server file structure. The user on the file server associated with the application has a data folder which holds all necessary application files. There can be any number of configuration files in the Configuration Files folder—the desired template can easily be picked in the configuration menu of the tablet application. Once the consultation form has been successfully completed and exported on the tablet, an SFTP connection will be established. If there is a valid network connection, the output documents will be sent to the Output Documents folder.

The Output Documents folder contains folders for each of the patient consultations, each containing a finalized DOCX file and images of handwriting results for verification. Each of these folders are uniquely identified by the date and time at which the consultation took place, so that patient information is not immediately divulged by simply listing the contents of the Output Documents folder.

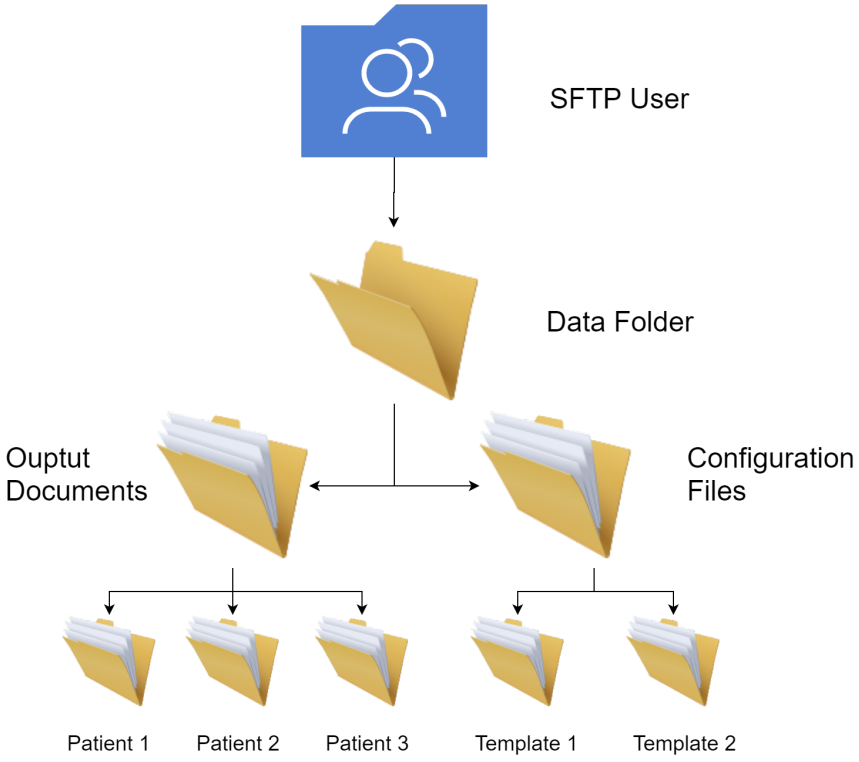

Figure 5.2: Server file structure

Each of the tablet-server interactions possible are depicted in Figure 5.3. When the tablet performs the "Get" operation, it requests two templates from the file server, a JavaScript Object Notation (JSON) file and a DOCX file. The JSON file is used to generate the form elements of the consultation onto the user interface of the tablet application. The DOCX file contains variables which are eventually replaced with the completed consultation form values. This completed DOCX file is subsequently sent back to the server using the "Put" operation, along with images of handwriting input in order to verify the consultation form data.

Lastly, "List" operation displays the names and dates of template and output files on the server as well as files that are still in the device's internal storage. This operation does not display any sensitive data—file names do not uniquely identify any patient information. This is done so that unauthorized users cannot simply gain

access to the tablet and immediately read patient names from files on the server and tablet. Additionally, file contents on the server and in internal storage cannot be read from the tablet—server file contents are not accessible and internal storage files are encrypted.

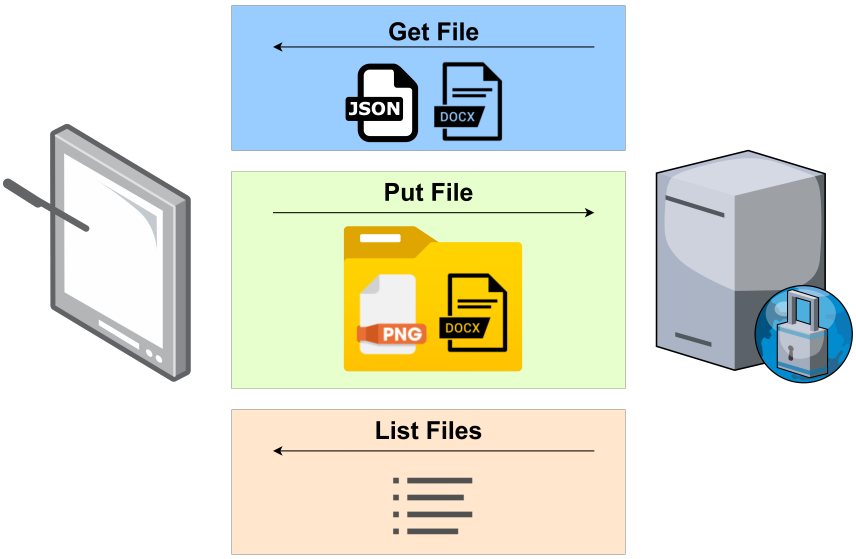

Figure 5.3: Tablet-server interactions

Chapter 6

Testing

6.1 Usability Testing

Our only user for this application, at least initially, is the neuropsychologist that the application has been developed for. The doctor does not know much about using technology or using mobile application interfaces, so the application has been designed in such a way that the doctor will need little to no training. To include some usability testing, the best method would be to watch people (preferably people who perform medical consultations) using our application who know nothing about it, although this may be somewhat difficult at this point in time due to health conditions in our area.

In the event we find people to try out our application in the future, it is important to watch their body language and ask them questions about their usage of the application, and about why they might have become confused using it, if they did at all. Our target audience for testing purposes would be individuals who have little to no experience in using mobile application interfaces.

6.2 Accessibility Testing

Although accessibility testing is important in many types of software applications, it is not considered a big priority for this project. The usage of the application is limited to individuals who would necessarily not have any difficulties manipulating a tablet device.

6.3 White-Box and Black-Box Testing

White-box testing is done by us as the developers of this project, as well as any programmer who we give access to our code and its modules. White-box testers can determine output and create their own test-cases based on a series of tests at every level (unit, system, integration). For black-box testing, we would have programmers and non-programmers try to see what the application does in all different cases, to possibly break it or cause unexpected results.

6.4 Conformance Testing

This application does not support conformance testing, as it is not required to meet a certain technical specification, standard, contract, or regulation. Although this is not required, some kind of manual testing may be done to see how the program conforms to the team's own chosen code standards. In addition, security testing could be done to determine application vulnerabilities should they exist. The developer page for Android lists relevant best practices when it comes to security.

First, credentials should be asked for before showing sensitive information. Our application doesn't have the ability to show human-readable documents stored on the device. All files containing patient information are stored in an encrypted state until they are transferred to the permanent file server. Even if the unlocked tablet is handed to someone else, the documents created by the app are not viewable. It is still necessary to protect the tablet with PIN/password/pattern/biometric credentials for HIPAA compliancy. Also included would be a quick sleep timer on the device, to prevent usage after a certain duration of inactivity.

Second, SFTP can be configured to use several different encryption algorithms, some of which are not considered HIPAA compliant. We will be configuring our connection to use SHA-256 encryption, which exceeds the HIPAA requirements. Transfers will be done on port 22, the default SSH port. This will not be vulnerable to the outside world because no port forwarding will be done for transfers over a local network.

Third, store data safely—data that may be stored from application usage will only be served to the server it's been sent to, and saved as a local encrypted copy. Storing encrypted data within internal storage will prevent other applications from reading the files, and if the app is ever uninstalled, the device will delete all the files that the app previously stored. Since the app may use some sort of initial document to populate a form, the validity of the document will be checked before performing any operations.

Lastly, keep services and dependencies up-to-date—to prevent any kind of application misconfigurations, errors, or security vulnerabilities, any dependencies will be updated as well as any system and security updates.

Chapter 7

Results

7.1 Realization of Requirements

We were successful in our endeavor to create an application that addressed all the critical requirements provided by our client. The requirement of easy creation and modification of template files was more difficult than we had planned for and was unfortunately unmet.

Security was always at the top of our priority list during development of this application, and there was never room for compromise. The HIPAA regulations regarding the storage of patient information are extremely important in digital systems—design decisions taken during development were always made with security in mind. All the research and planning that went into securing our application gives us confidence to say that this project meets all the requirements in order to be HIPAA compliant and employed in a medical context.

Seamless integration into the client's office environment is necessary for this project's success. Our application is designed to improve workflow greatly by digitizing paper forms. Fortunately, we were able to make our application work alongside the office's existing secure file server. This means that employees will not have to change the way they access completed consultation documents. Minimizing the amount of workflow adjustment required for our client's office staff was key to integrating our application into their environment.

Our research into finding a solution for our client to easily edit JSON configuration file was not successful. We reviewed many potential libraries and APIs; unfortunately, we were unable to find anything that would be simple for a non-technical person like our client to utilize. We were, however, able to validate the configuration file using a JSON schema. Using online validation tools, it is simple to find syntax and type-mismatch errors in the formatting of the JSON input file—this should minimize the amount of time required to maintain it.

It was difficult to find a solution that allowed the values entered into the application's form to be inserted into a Word document in a presentable way. Our first approach was to programmatically define formatting for each input type on the form. This method produced output documents that were unsightly and difficult to read. One key requirement for the project is that the output document needs to present patients' information in a format that is quick and easy to read for other medical professionals. In order to create a flexible system that can format documents dynamically, we made use of docx4j's variable replacement functionality.

This allows the client to create a Word document with their desired formatting to be used as a template for form values to be inserted into. This solution creates an extremely flexible way to format the output document at the expense of initial setup time.

7.2 Realization of Standards and Constraints

We were able to fully adhere to the standards and constraints that were defined for this project. The utilization of SFTP for file transfers ensures that any system is capable of transferring and hosting the files from our application. Our application should be portable to newer versions of Android and the devices that run it. We were limited in the range of input element types available on the tablet's form due to the way they were designed in BuildFormer. The source code from the library could have been modified to allow for additional input options; for the sake of time and simplicity, we avoided changing the BuildFormer source code.

7.3 Testing Results

Due to time constrains for this project the amount and type of testing for this project was somewhat limited. A majority of the testing conducted on our application was white-box testing. It is not ideal for the developers on a project to do all of the application testing since the test scenarios used can be biased based on the knowledge of the applications inner workings. It has been difficult to meet often with our client with current health concerns and both of our busy schedules. We would have liked to do more usability testing with our client during development. Unfortunately, some parts of our application took longer to develop than anticipated. This lead to parts of the project being in an unstable state while development continued. This unstable state, mainly a lack of integration between the form filling and output document generation functionality, meant that the result of filling out the digital form were not immediately available for review. Another setback came from the requirement of our application to fetch template files from an SFTP server implementation. This requirement means that a fairly complex test environment is required fully test functionality.

Even with all the setbacks and unfortunate circumstances we were successful in fully testing the main functionality of our application.

Chapter 8

Future Work

Future work on the application could include improvements to consultation form template customization. Currently, the process to modify consultation form templates can be somewhat time-consuming. Both the Word DOCX file to apply formatting and the JSON file need to be updated, and the objects that represent form elements in the JSON file must match up with the DOCX file variables in order for the application to function. Our application makes sure that both configuration files contain the same variables, but this process could be improved with a more efficient design. Unfortunately, any improvements to the current system would require much more complexity and time which we couldn't afford.

Another improvement could be increasing the number of form input objects available in the digitized form. Due to limitations in the BuildFormer source code, we were unable to include all available form elements in the final application. With additional time, it would be possible to modify the project's source code to include identifiers needed for inserting form values into the DOCX output file. For example, two of the input options that had to be left out were a date picker and a time picker. These input elements would make it easier to enter dates and times into the digitized form.

One example of a better input file configuration system would be a graphical user interface that allowed form elements to be simply dragged and dropped into place and edited. This would have addressed the unmet requirement that arose during development of the application—our client would most easily be able to modify template files by not having to edit raw JSON or DOCX files, where simple typos could result in syntax errors. Another idea, while difficult in terms of software development, would be to automatically generate the template file by analyzing the client's current consultation form either through image processing or file parsing with a machine learning model.

However, the initial transformation of a client's consultation form to one that can be utilized by the application will always be inefficient, because every type of medical professional uses consultation forms that are structurally opinionated and differ from field to field. This issue could be solved if there were national or global standards in place for the structural conventions of consultation forms in the field of medicine.

The portability and security of the application could also be improved in the area of handwriting recognition. Currently, handwriting recognition relies on the usage of the third-party library MyScript. With enough time and added complexity, a custom

handwriting recognition system could be built into the application. While requiring more application resources, this built-in solution would eliminate a software library dependency, which is always a good thing when it comes to application security. Removal of the MyScript library would also eliminate the need for the application to contain a certificate verifying the validity of the MyScript account being used to perform handwriting recognition requests.

Chapter 9

Conclusion

As we have stated, the medical industry is quickly transitioning towards the use of Electronic Health Record systems to replace paper documents in offices across the nation. Unfortunately, small medical practices have many obstacles to overcome when it comes to incorporating EHR systems into their work environment. The most prominent issue is the cost of these systems. Significant upfront costs and monthly maintenance fees of hundreds of dollars makes implementing a traditional EHR system very unappealing. Our goal for this project was to create an open source solution to digitizing paper forms that addresses all the security requirements put in place by Health Insurance Portability and Accountability Act (HIPAA).

A key requirement for our application was the ability for our client to have more interpersonal interaction while doing consultations than a traditional EHR system's keyboard-centric approach to form filling would allow. Our research concluded that using a tablet with a stylus would allow our client to still hand write assessments of patients' conditions instead of typing them. The handwriting entered into the application is converted to text values that can be later proofread and finalized office staff. The handwriting recognition capabilities present in our application are implemented using an API from the company. This is a proprietary piece of software that provides a user interface and excellent handwriting conversion tools. With significantly more time spent developing this application, it would be possible to eliminate all monetary cost for this project by developing built-in handwriting recognition technology, save for the cost of a tablet to run the application. This critical system eliminates the need for a transcriptionist to tediously enter written assessments into the final consultation document. To further improve our client's transition to using our application, we worked with the existing infrastructure present in their office. Our goal is to minimize the amount of workflow adjustment required for our client's staff when using our application.

The most critical requirement for this project is the security of patient information. HIPAA gives well defined guidelines for the handling medical information. These rules and guidelines are not specific in the level of security required by applications handling patient information. The reason for this is the rate at which technology is moving. It would not be wise to create specific requirements that may be quickly outdated. The encryption methods that are used in this application are the industry standard for protecting data. The AES-256 encryption algorithm has over a vigintillion different combinations. These encryption standards would take millions of years to crack on the fastest supercomputers in the world. The server

communication protocol SFTP used in this application is widely used and known to be secure, with AES-256 encryption protecting every transmission. SFTP can be installed on almost any system, which ensures our application can be used with most—if not all—file server implementations. Unfortunately, many medical practices may not have an existing file server in their office environment. This will limit how widely our application can be used in the medical industry.

The largest setback for our application is setup time. The template files required by our application to generate a digital form and output a finished word document take a considerable amount of time to create. Fortunately, these documents only need to be created once. The time saved from transitioning away from paper consultation forms easily makes up for the initial setup requirement. It is possible for future development on this project to help streamline the process of creating template files. The benefit of using template files to create the digital form and resulting consultation document is the flexibility to create any type of digital form needed for an arbitrary number of use cases. There is room for future development to increase the number of input options available in the digital form generated from template files. With enough time, the source code from the library could be modified to add additional input types—more work would need to be done to update the functions used to generate the output Word document.

Chapter 10

References

[1] Material Components. *Material-Components-Android*. 2020. URL: https://github.com/material-components/material-components-android.

[2] Markus Friedl and Damien Miller. *OpenSSH*. 2000. URL: https://www.openssh.com/.

[3] Google. *Gson*. 2020. URL: https://github.com/google/gson.

[4] Jason Harrop. *docx4j*. 2020. URL: https://github.com/plutext/docx4j.

[5] JCraft Inc. *JCraft JSch*. 2018. URL: http://www.jcraft.com/jsch/.

[6] MyScript Inc. *MyScript*. 2020. URL: https://developer.myscript.com/.

[7] de Jong. *Json Editor Online*. 2020. URL: https://github.com/josdejong/jsoneditor.

[8] Nick Maynard. *JSON Schema Lint*. 2020. URL: https://github.com/nickcmaynard/jsonschemalint.

[9] James Newton-King. *Json.NET Schema*. 2020. URL: https://github.com/JamesNK/Newtonsoft.Json.Schema.

[10] Girish Raman. *BuildFormer*. 2017. URL: https://github.com/RGirish/Build-Former.

[11] JSON Schema. *JSON Schema*. 2020. URL: https://github.com/json-schema-org/json-schema-spec.

Appendix A

Development Costs

A.1 Institution Costs

Western Michigan University did not take on any costs during the development of the application.

A.2 Sponsor Costs

Our client purchased three (3) Samsung Galaxy Tab S6 tablets for a total of approximately $1200. The tablets were loaned to the team for the duration of application development and subsequently returned on completion.

A.3 Team Costs

The team did not take on any costs during the development of the application.