12-15-2021

Utilizing Software Engineering and Cloud Computing Principles to Develop the Revised Self-Report Assessment of Functional Visual Performance (R-SRAFVP) Application

Kirk Hedlich
Grand Valley State University

Utilizing Software Engineering and Cloud Computing Principles to Develop the Revised Self-Report Assessment of Functional Visual Performance (R-SRAFVP) Application

Kirk Hedlich

A Project Submitted to

GRAND VALLEY STATE UNIVERSITY

In

Partial Fulfillment of the Requirements

For the Degree of

Master of Science in Computer Information Systems

School of Computing

December 2021

GRAND VALLEY
STATE UNIVERSITY

The signatures of the individuals below indicate that they have read and approved the project of Kirk Hedlich in partial fulfillment of the requirements for the degree of Master of Science in Computer Information Systems.

[illegible]	12-14-21
Dr. Byron DeVries, Project Advisor	Date
[illegible]	12/15/2021
Dr. Paul Leidig, Graduate Program Director	Date
D Robert Adams	12/15/2021
Dr. Robert Adams, ACS Program Chair	Date

Table of Contents

Table of Figures

Abstract

Can principles from software engineering and concepts from cloud computing be applied to and aid in the development of a small project, specifically improving the use of the Revised Self-Report Assessment of Functional Visual Performance (R-SRAFVP)? The target for this project is to create a new application and improve on existing attempts to move the R-SRAFVP assessment from an electronic document format to a web-based application. The new application should provide better ease of use, simplicity in design and understanding for use, and hopefully increased access and adoption by Occupation Therapists who specialize in low vision rehabilitation. The benefits of using cloud computing would be to reduce costs (e.g., website vs. stores), implement scalable resources, reduce technical complexity (e.g., single code base for all device platforms, single deployment for production, etc.), enable cross-platform use (same source for all device platforms for use) and reuse cloud services instead of having to re-develop them. The discipline for the project is to show how using software engineering concepts engages the stakeholders, confirms requirements, forces design reviews, aids in the reduction of development rework, establishes the need for testing and, hopefully, mitigates surprises found in small projects not using software engineering concepts.

Introduction

In healthcare, obtaining the right treatment for the right condition is paramount. The costs associated to healthcare are ever increasing which means having a way to quickly assess a patient's condition helps to reduce costs and improve the direction for treatment. When dealing with the five human senses - touch, taste, smell, hearing, and sight – loss or lack of sight has a significant impact on a person's experiences and quality of life. There are many medical professionals working with patients with low vision, one of these is Occupational Therapists (OT) who specialize in low vision rehabilitation. OT's use a variety of tools to help patients regain their lives with adaptation, but how does an OT assess a patient's low vision? Dr. Mary Warren, from the University of Alabama, Birmingham created the Revised Self-Report Assessment of Functional Visual Performance (R-SRAFVP) ("R-SRAFVP Toolkit").

What is the R-SRAFVP assessment and why is it important? The R-SRAFVP assessment is used by OT's who specialize in low vision to assess patients functional vision performance. The assessment reviews how well a patient manages daily or routine activities that are dependent on vision. The assessment covers eight categories of evaluation. Each category contains several questions ranging from dealing with medications/food or personal grooming to reading, writing, and getting around without issue (e.g., tripping, going up or down stairs, etc.). Each question uses the same scale for answers to standardize the responses and be able to provide a common standard impairment score. Based on how each question is answered, the assessment helps to gauge the level of dependence or independence of the patient. The aggregate score is used to determine a patient’s percentage of impairment and a Medicare G-code required for treatment.

The R-SRAFVP was released in 2017 and comes in three forms: Microsoft Word document (“R-SRAFVP”), a Microsoft Excel spreadsheet (“R-SRAFVP”), or an internet accessible version of the Microsoft Excel spreadsheet within a wrapper application as R-SRAFVP App (Mohiuddin). Although each of these forms work on their own, reducing human error, ease of use, simplicity of design, greater accessibility and additional functionality would be helpful to improve use/adoption for the assessment. The case can be made that the Microsoft Word version is straight forward but contains no automation and means the user must manually score the assessment. Because the assessment is manually calculated, the assessment is prone to human error. The Excel version reduces calculation errors by removing the manual calculation and introducing automated calculations. The problem with both versions of the assessment, a user must have either Microsoft Word or Excel on their device when completing the assessment with a patient. The R-SRAFVP App works toward ease of use and greater accessibility by wrapping the Microsoft Excel spreadsheet in a shell application and making it available via the internet. This version makes the assessment available on the internet but lacks much of the experience and functionality most internet applications contain. In addition, users have come to expect quite a bit from applications and their online experience.

Although the population of users for the R-SRAFVP assessment is small, creating a revised version of the R-SRAFVP App as a true internet application is an opportunity for learning and improving on the existing implementations. The updated version should provide better ease of use (mobility and cross platform support), simplicity in design (focus on implementing the assessment and ease of scoring and export functionality), and increased access for users (internet access without wrappers). This can be achieved with a mobile first focus using the benefits of using cloud computing. In addition, cloud computing would help to reduce costs (e.g., website vs. application stores from Microsoft, Google, Apple, etc.). Implementing cloud computing would also improve the technology side by utilizing scalable resources, reduce technical complexity (e.g., single code base for all device platforms, single deployment for production, etc.), enable cross-platform use (same source for all device platforms for use) and reuse cloud services instead of having to re-develop them.

Development takes discipline and with such a short timeframe, the project will show how using software engineering helps to achieve a quality outcome. By using software engineering, the project will engage stakeholders, confirm requirements, forces design reviews, aid in the reduction of development rework, establish the need for testing and, hopefully, mitigate surprises found in small projects not using the discipline of software engineering. Through this project, this author hopes to answer the question, can principles from software engineering and concepts from cloud computing be applied to and effectively complete the development of a small project and improve the Revised Self-Report Assessment of Functional Visual Performance (R-SRAFVP) as an internet application.

Project Management

Discipline for the project comes from incorporating software engineering principles and using the Incremental development model (“Software Engineering: Incremental Process Model“). Because of the size of the project and availability of reviewers, the Incremental development model was the best choice to define the requirements and design upfront while creating successive versions to build out the application until all requirements and designs are realized. This proved to be the most applicable and useful as it allowed for complete planning, requirements definition and design before development work started. Then development work would be done for fully functional releases that incrementally built up the application to fulfill the requirements. By using this approach, about half of the project time was spent on requirements, design definitions and reviews before any development work started.

By focusing so heavily on the define and design phases for the project, the intended goal was that the foundation for the development would be quite solid and provide for easier understanding for what needs to be development. The focus was to reduce the problems usually encountered with requirements and design to a minimum. The goal was to make the technical implementation the bulk of the project’s challenge rather than having to manage define and design during the development phase of the project. In addition, assessing each release for validation of requirements and verification of design would be straight forward.

Development Process

The software development process used for the project was a simple Incremental model. To track progress a simple WBS was used to plan and assign work to the weeks in the semester (see Timeline section for details). This provided the following benefits: 1) a simple means to track progress and 2) a checklist to determine if deliverables were complete.

Reviews for requirements and designs were conducted via email with deadline dates for feedback. Reviews were done in parallel with new work. This means as drafted versions of requirements were done and sent out for review, design work progressed in parallel while waiting on feedback reviews. This was a calculated risk as feedback could have come back with enough changes causing a large amount of rework. This did not occur even though the risk was not mitigated.

Separate Project Plan and Progress Report documents were used to document the plan and track progress to plan based on deliverables. Both are available in the project's Github repository (see Appendix for details).

Deliverables

The following software engineering deliverables were completed as part of the project. These are available in the project's Github repository (see Appendix for details).

- R-SRAFVP History document – prior project work summary
- R-SRAFVP Project Plan
- R-SRAFVP Progress Report

- R-SRAFVP Use Case Diagrams
- R-SRAFVP Use Case Specification
- R-SRAFVP Requirements Specification
- R-SRAFVP Requirements Traceability Matrix
- UI/UX Design Model
- UI/UX Wireframe Model
- UI/UX Application Preview (mock model for walkthrough experience)
- Requirements and Design review feedback
- R-SRAFVP System Architecture Model
- Code under version control (Github)
- Testing Specification

Releases

Development work was planned and divided into three releasable phases. Due to time, the project releases were scaled back to two, Release 1 and 2. The Reflections section discusses this in more detail. Listed here are the initial planned versions. The released version can be viewed at www.rsrafvp.com (see Appendix for details).

- Release 1 – a usable assessment without advanced features and a fully framed site,
- Release 2 – advancing the assessment features for authenticated users,
- Release 3 – incorporating functionality to do research on the data being accumulated to improve the R-SRAFVP assessment.

Used/Implemented Technologies

The following technologies were used to document the definition, design, or development of the R-SRAFVP application.

- Microsoft Office for Word, Excel, and PowerPoint
- Github (version control and repository for all things R-SRAFVP related)
- Bootstrap Studio - for user interface design work (HTML/CSS)
- Microsoft Visual Studio Code – JavaScript work, production deployments
- HTML, CSS coding languages - for user interface
- JavaScript language – for page/site functionality
- Firebase for authentication (basic and additional providers), database, and analytics services
- Google® for website hosting services
- Google® authentication service as additional provider

Anticipated Problems

The following list describes the initial thoughts for anticipated problems that may come up during the project work. The list is unaltered from the beginning of the project to provide a basis for the Reflection section for this project.

- Under-planning the number and types of assignments
- Being too optimistic for assignments to get the work done
- Technical challenges with tools, technology, and/or development

- Scope creep to improve functionality and capability before delivering a working tool
- Too many assignments to cover in semester, what could be cut out.
- Testing will be cut short

Timeline

The Timeline defines the entire project effort into major phases with the weeks representing when the work needs to be completed. A separate Progress Report document was used to document tracking progress to plan based on deliverables. The Progress Report is available in the project's Github repository (see Appendix for details).

- Planning/Definition (semester weeks 1-3): Complete the project plan, R-SRAFVP history, use cases, and requirements deliverables.
- Design (semester weeks 4-6): Complete the activity diagrams for user flows, start the traceability matrix with mapping requirements to initial deliverables, develop UI/UX design and user flows, and the initial work for prototyping of the PWA, potentially testing technologies on desktop, tablet, and phone devices.
- Development Release 1 (semester weeks 7-10): Complete the system architecture, the development of the PWA, testing on different platforms for all planned testing types.
- Deploy (semester week 10): deploy project to cloud platform for public use and feedback.

- Development Release 2 (semester weeks 11-13): Development of the logged user functionality, testing on different platforms for all planned testing types.
- Deploy (semester weeks 13): deploy project to cloud platform for public use and feedback.
- Debug (semester weeks 11-14): resolve production issues.
- Presentation (semester weeks 12-14): Complete the project report, project presentation for final submission by the last week.
- Potential buffer (week 15)

Organization

The R-SRAFVP application is a progressive web application (PWA) rather than a typical website. A PWA blends the capabilities of native platform applications with standard web applications to gain the benefits of both while reducing the negatives from both. What makes a web application a PWA?

- *Discoverable, so the contents can be found through search engines.*
- *Installable, so it can be available on the device's home screen or app launcher.*
- *Linkable, so you can share it by sending a URL.*
- *Network independent, so it works offline or with a poor network connection.*
- *Progressively enhanced, so it's still usable on a basic level on older browsers, but fully-functional on the latest ones.*
- *Re-engageable, so it's able to send notifications whenever there's new content available.*
- *Responsively designed, so it's usable on any device with a screen and a browser—mobile phones, tablets, laptops, TVs, refrigerators, etc.*
- *Secure, so the connections between the user, the app, and your server are secured against any third parties trying to get access to sensitive data.*

(“Introduction to Progressive Web Apps - Progressive Web Apps (PWAs): MDN”)

Although the key elements for a PWA seems like additional work, many of these elements need to incorporate into native platform applications (e.g., Android®, iOS® or Chrome® applications) and standard web applications. This project is no different and resolves the native application issue by being a PWA and incorporating as much cloud computing functionality as possible. The R-SRAFVP Architecture Diagram (Figure 1) shows how Firebase® services and external services/sites are used to combine cloud computing and traditional web links.

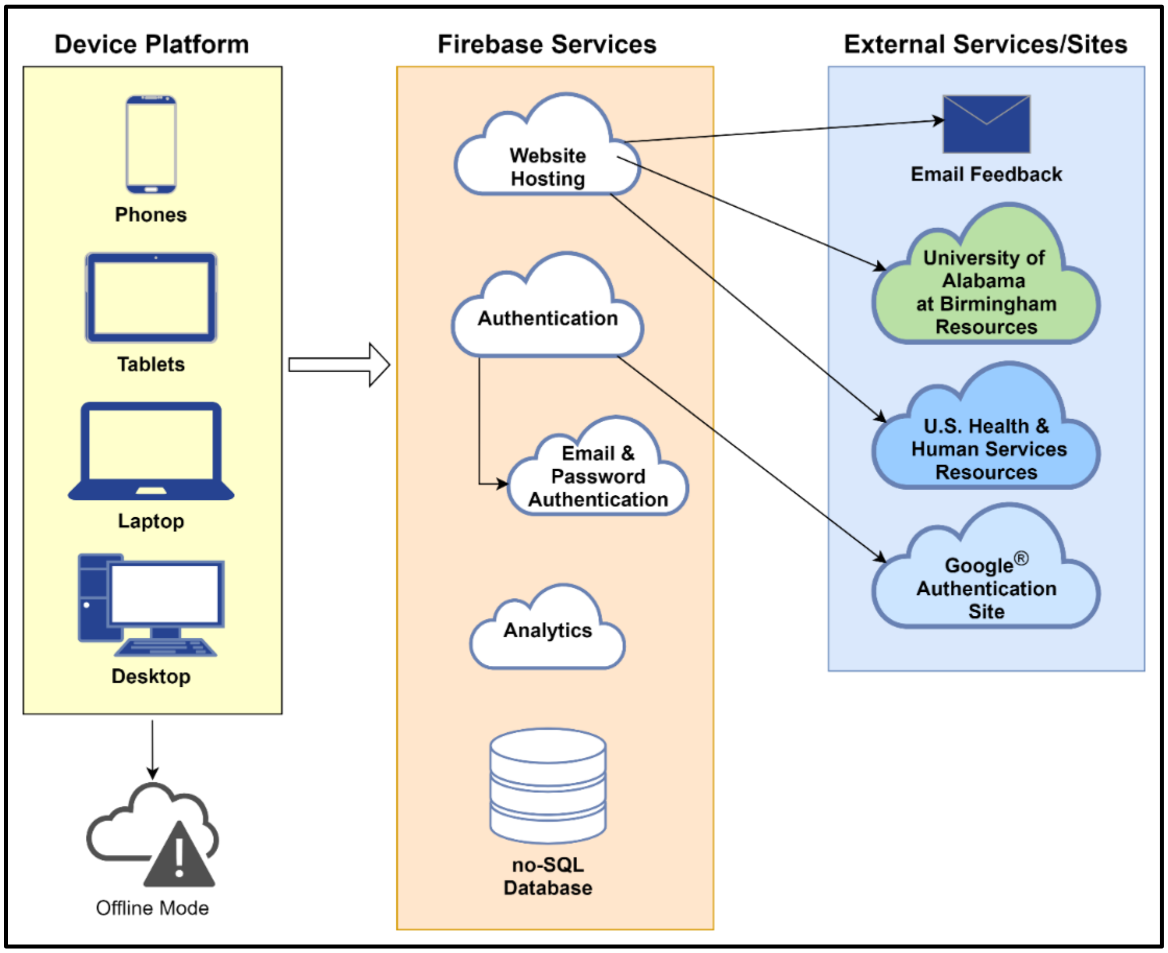


Figure 1: R-SRAFVP Architecture Diagram

The general architecture for the R-SRAFVP application is mobile first approach and using several cloud computing services through a single provider. By using this approach, all services utilized by the application are in the software-as-a-service (SaaS) implementation. This approach enables the project to expand as needed (e.g., when usage increases, so do resources supporting the application dynamically), only pay for what is used and implementing services as the application itself is further developed. The R-SRAFVP Architecture diagram (Figure 1) shows the high-level diagram for how the user interacts with the application (via device platform), what types of services are being utilized through Firebase® and what services/sites are external to Firebase®. The general flow for service utilization is service work for online/offline access, analytics service, authentication service and database service.

The first service is a service worker which is utilized when the user first accesses the site's URL. The service worker determines if the user's device is online, to access all online services, or offline, meaning the application will use cached content from the user's device. Caching of content is done the first time the user can access the site, from that point forward the cached content is only refreshed as the cached content is changed.

Assuming the user is online, the application has two primary user modes: guest mode or logged in user mode. Guest mode does not use any additional services, but to verify a user the application uses the Firebase® authentication service. By using this service, users can create a Firebase® account and then login using their existing account. In addition, the Firebase® authentication service provides a way to use additional providers for user login verification to minimize work and use external pre-built services. For the R-SRAFVP application the only additional authentication service implemented is for users with Google® accounts.

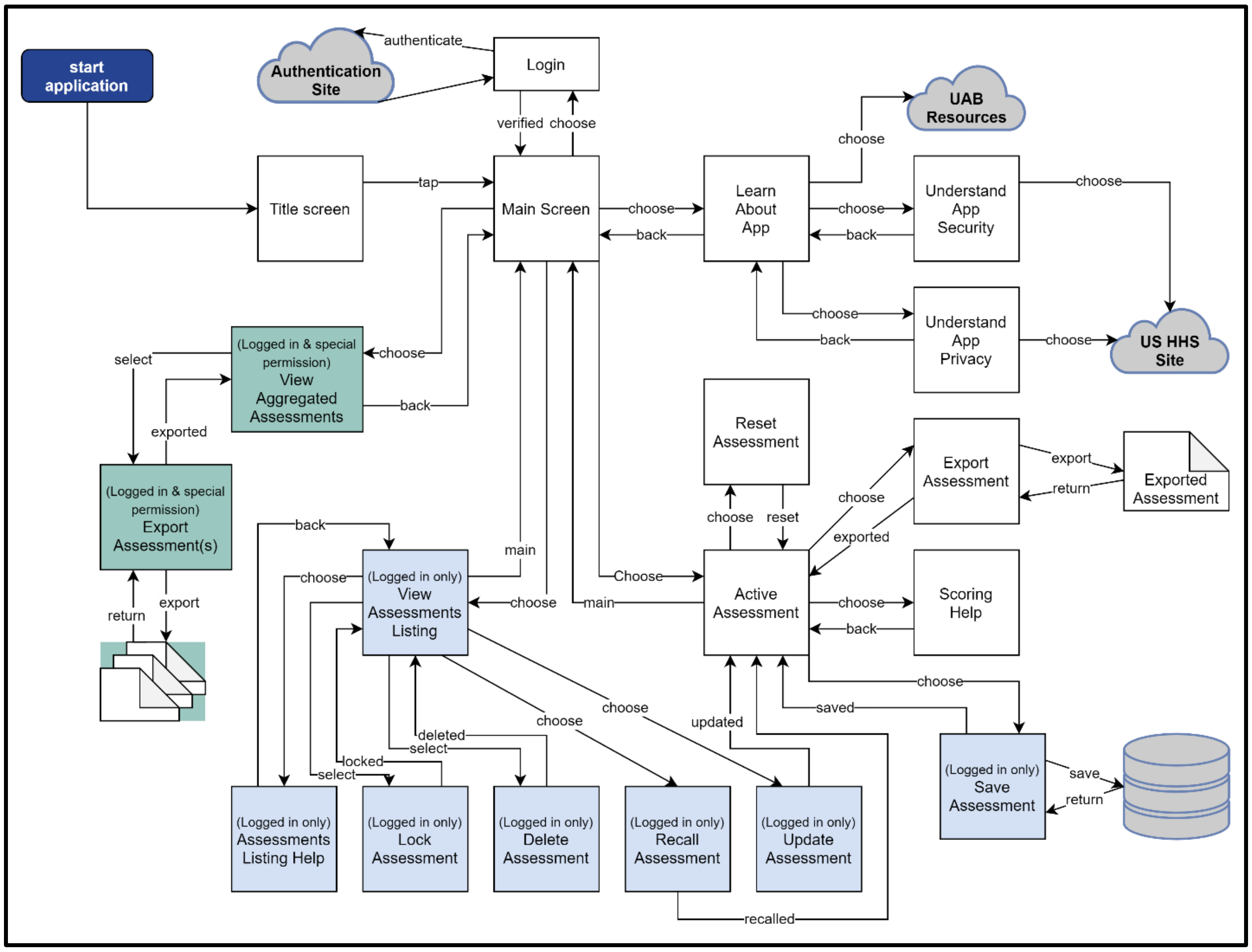
Authenticated users get additional functionality for their assessments. Users can save, recall, update or delete previously saved assessments. To store saved assessments Firebase® Firestore, a NoSQL database, is implemented. Again, by using a pre-built service, this increases development speed and minimizes custom development work.

With the application being a medical assessment and web-based version of the R-SRAFVP electronic document, users needed access to three specific links for reference: the R-SRAFVP Toolkit (“R-SRAFVP Toolkit“), the United States Department of Health and Human Services (US HHS) for the Privacy (“The HIPAA Privacy Rule”) and Security (“The HIPAA Security Rule”) rules. When users have questions about why to use it or how to use it, users need to reference the R-SRAFVP Toolkit (“R-SRAFVP Toolkit”) website for more information. Then there is the issue of compliance as set forth by the United States Department of Health and Human Services for medical applications. Users wanting to know how the R-SRAFVP application complies with Privacy (“The HIPAA Privacy Rule”) and Security (“The HIPAA Security Rule”) rules need links to the original information from the US HHS for both. All three links are standard web links and provide greater knowledge for a user’s understanding.

When users encounter a problem or have questions about the application, many websites provide a means for feedback. The R-SRAFVP application is no different. An email service is incorporated to provide users a means with which to provide feedback. Any feedback is aggregated to an admin account for further review.

Finally, in addition to user-based services, Firebase® Analytics service is used to understand site usage and user behaviors. Currently standard site metrics are tracked through the Analytics wrapper service and available for use and viewing through the Firebase® R-SRAFVP project admin console.

Once the user accesses the website, it works like a typical website. The R-SRAFVP User Interface Architecture Diagram (Figure 2) shows a high-level overview for the different pages of the site and how those pages are linked. In addition, the different pages are color coated for the different user modes and which user types have access to which pages. The white pages show guest mode accessibility. The blue pages show accessibility for a logged in user. The green pages show special user accessibility that was planned to be done as part of this project but will now be part of a future release instead. The user interface has been developed using HTML/CSS pages with specific transitions happening to move the user from page to page. JavaScript is used for each page for custom functionality.


start application
Authentication Site
authenticate
Login
verified
choose
UAB Resources
Title screen
tap
Main Screen
choose
back
Learn About App
Understand App Security
Understand App Privacy
US HHS Site
(Logged in & special permission) View Aggregated Assessments
select
exported
(Logged in & special permission) Export Assessment(s)
return
export
Reset Assessment
reset
Export Assessment
Exported Assessment
exported
Active Assessment
Scoring Help
main
Choose
(Logged in only) View Assessments Listing
deleted
select
locked
updated
saved
(Logged in only) Save Assessment
save
(Logged in only) Assessments Listing Help
(Logged in only) Lock Assessment
(Logged in only) Delete Assessment
(Logged in only) Recall Assessment
(Logged in only) Update Assessment
recalled
Figure 2: R-SRAFVP User Interface Architecture Diagram

To start, all users move through the Title Screen to the Main Screen which acts as the primary landing page for all users. From there, most users will work on an active assessment and move to the Active Assessment page. Only one active assessment is allowed at any given time. This restriction helps with both focus for the user and allowing for Guest users to use the application without authentication. When users are brought to the active assessment screen, the assessment is reset and ready to be filled in. Beyond the active assessment, Guest users have limited functionality. Users who log into the application extend their functionality with the ability to save the active assessment or recall / update / lock / delete previously saved assessments from their own usage. The scope of the application covers the following functionality:

- Assessments fields and data must follow the R-SRAFVP Form (R-SRAFVP Toolkit, n.d.) for question flow, question scoring and output like the form.
- Users who do not authenticate (login) into the application will be limited to creating, exporting, and resetting assessments. Additional functionality will require users to authenticate (login) into the application.
- Users who do authenticate into the application will have access to extended functionality for saving, view lists of saved assessments, recalling, updating, locking, and deleting assessments.
- Authenticated users will only have access to their own assessments.
- Access between users for assessments will be prevented (ex: reassigning assignments between users or sharing of assessments for viewing from within the application).
- Assessment data (i.e., patient characteristic data, question answers and scoring data) generated by application users will not be tied to users or patients identifying data to mitigate HIPAA concerns for compliance and privacy.

- Any identifying users or patient data will be minimized and anonymized as much as possible to mitigate HIPAA concerns for compliance and privacy.
- The assessment data gathered from all users is intended for long term review for improvement of the R-SRAFVP itself by special permission and for researchers only.

Reflection

By using the discipline and deliverables from software engineering, the project was shaped into a usable application. The process provided an understanding for what needed to be developed and when deliverables would be ready. The process allowed for confirmation and correction by stakeholders on each deliverable up to development. It provided for communication about timelines and deliverables. Although use cases required effort to complete, additional requirements were elicited because of things not realized during requirements definition. The user interface design, wireframe and application preview helped to visualize the implementation of the requirements and use cases for the stakeholders. Minor adjustments were made because of the visualizations. Also, by using the Iterative model, the discipline forced a serial completion of requirements, design and usability were correct before any development work was started. This made sure the needs and wants of the stakeholders were addressed, the basis for the development work was set.

By having the requirements and user interface designs already done, implementation questions were reduced. Again, it seems like a great deal of time was spent not developing the application, but it shows why this important for the implementation. Rather than questioning how something should work or designing it at the time of implementation, development work was done using the user interface design, use cases or requirements as references instead. This reduced the typical development lag for questions and answers. For this small project and short timeline, any lag could be very impactful.

In general, the use of software engineering principles and deliverables fared better than on most projects, but some difficult choices were made to reduce scope to make the delivery deadline. The use cases, requirements and user interface designs were documented, reviewed, and finalized by external reviewers. This an important note since turn-around in such a short timeframe tends to be dropped in favor of minimal requirements and the attitude of "just get it done." The main impacts were to diagrams and models (i.e., Activity diagrams, Class diagram, Sequence diagrams). The intent was to produce these to provide a better defined and designed project release. The plan was to produce Activity and Class diagrams as part of the project, but technology challenges proved to be greater and needed to be tackled to complete the project.

By incorporating cloud computing concepts, the velocity of delivery continued to improve. Cloud computing services helped to reduce development time and effort while increasing the capabilities without a large investment. By focusing on a PWA and mobile first approach, several cloud computing principles were incorporated into the final project releases.

- The application is a PWA and installable, works both online and offline,
- The application uses a responsive display design to allow different device types and display sizes to view and use the application
- The application is a website and uses Search Engine Optimizations (SEO) to make it more discoverable,
- The application can be easily shared as a URL,

- The application was updated twice. Users automatically took advantage of the new functionality without needing to update or install anything,
- The application has incorporated security to differentiate guest and authenticated user functionality as well as database security.

There were some significant technical challenges throughout the course of the semester for this project. All the technical challenges were reduced to three main points:

- ○ Using three new languages for the project with no development experience (HTML, CSS, JavaScript)
- ○ Creating HTML pages through manual coding instead of using UI/UX design tool.
- ○ Incorporating Firebase services

Two main problems surfaced while using the new languages, manually creating the user interface and customized functions. To be able to create HTML pages that showed like the UI/UX design and code them by hand was a time sink. After a brief evaluation period, a tool was found to expedite the HTML/CSS development, Bootstrap Studio. It helped to design the webpages, place UI elements and provided a way to preview the design without constantly deploying prototype code. Bootstrap Studio is an Integrated Development Environments (IDEs), reducing the amount of tedious hand-coding needed for HTML/CSS. With the requirements and UI designs to use for reference, creating the overall site structure and interactions took much less time than when trying to define, design and develop all at the same time. Focus was then turned to the use of JavaScript for custom functions. In general, this is straight-forward but for a first-time developer, understanding the differences for some the subtle nuances was again a time sink. Luckily, there are some good websites to help explain JavaScript syntax and provide examples for good and bad implementation. Along with some playground sites for experimentation, technical issues using JavaScript were overcome.

The biggest hurdle was incorporating the Firebase services at all. There were subtle differences between using their web SDK 8 and 9 versions. This was not well understood and required a great deal of research and trial development time until the first Firebase service was incorporated. By incorporating Firebase services, metrics for site analytics was now working. The significance was the ability to see what was happening on the site and have information about where people were using it or the types of devices connecting to the site.

The next challenge was dealing with third-party authentication. Enabling third-party authentication required understanding and setup of third-party provider accounts and authentication for third-party integrations. Google was the only third-party authenticator planned for initial release. Facebook and Microsoft third-party authentication were attempted but deferred for a later release due to difficulty in getting them to work and time again was slipping away. The good news is that simple authentication and Google authentication were now functioning.

The last major element for the project was Firebase database connectivity. Once the database was connected, the hope was that save, retrieving and manipulating records would be easier. In hindsight, Database connectivity proved to be one service that was easier than expected.

Although the initial plan was to create three releases for the new application, this was a partial success. Release one was deployed to production on Nov. 1, 2021. By using the Iterative model, it was the heaviest workload from the software engineering principles viewpoint. Release 1 required all use cases, requirements, and design work to be done prior to development. In addition to software engineering deliverables and stakeholder reviews, Release 1 also required new programming language learning/understanding, evaluation of tools and review of language libraries, this was quite the achievement. By having the basis for the application done, using software engineering principles truly helped when faced with all additional challenges to get Release 1 completed. The interesting thing is how much faster Release 2 occurred. Although Release 2 is not fully realized, the base functionality was release 20 days after Release 1 and went to production on Nov. 20, 2021.

As with all software projects, this one is not done. The roadmap for the R-SRAFVP application is short but it will take a little more effort to complete.

- Complete any remaining functionality for Release 2
- Complete all functionality for Release 3
- Add more authentication providers (e.g., Microsoft or Facebook)
- Reducing some of the technical debt created during this project
- Simplifying the user workflows
- Incorporate automated testing

Was the project successful? The goal for this project was to create an application with the ability for users to generate individual assessments for patients. This was achieved! Using Firebase Analytics, the figure shows the site metrics for November 2021. The R-SRAFVP User Activity Diagram (Figure 3) shows how the project continues to gain users over time. The R-SRAFVP Usage Location Diagram

(Figure 4) shows the locations of users using the application. The R-SRAFVP Events Diagram (Figure 5) shows the diverse types of events being gathered. As a note, “page_view” is equivalent to the number of HTML pages that have been viewed on the site. Although there are a number of metrics available, the purpose of these is to show that the site is live and is being used.

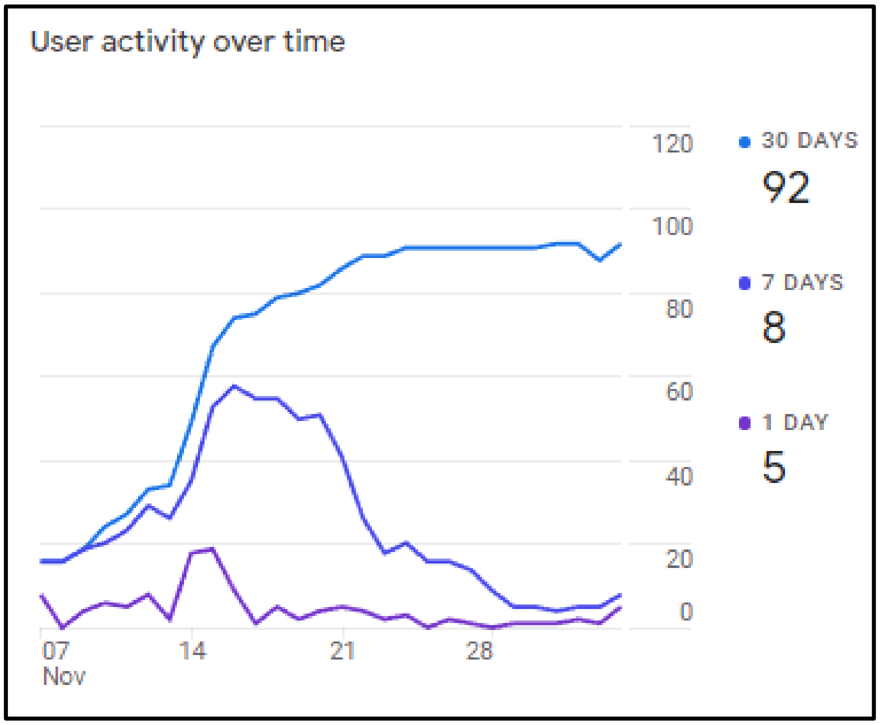
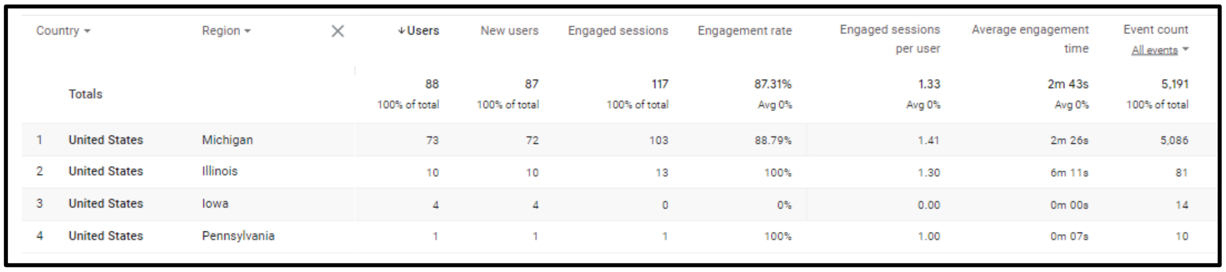
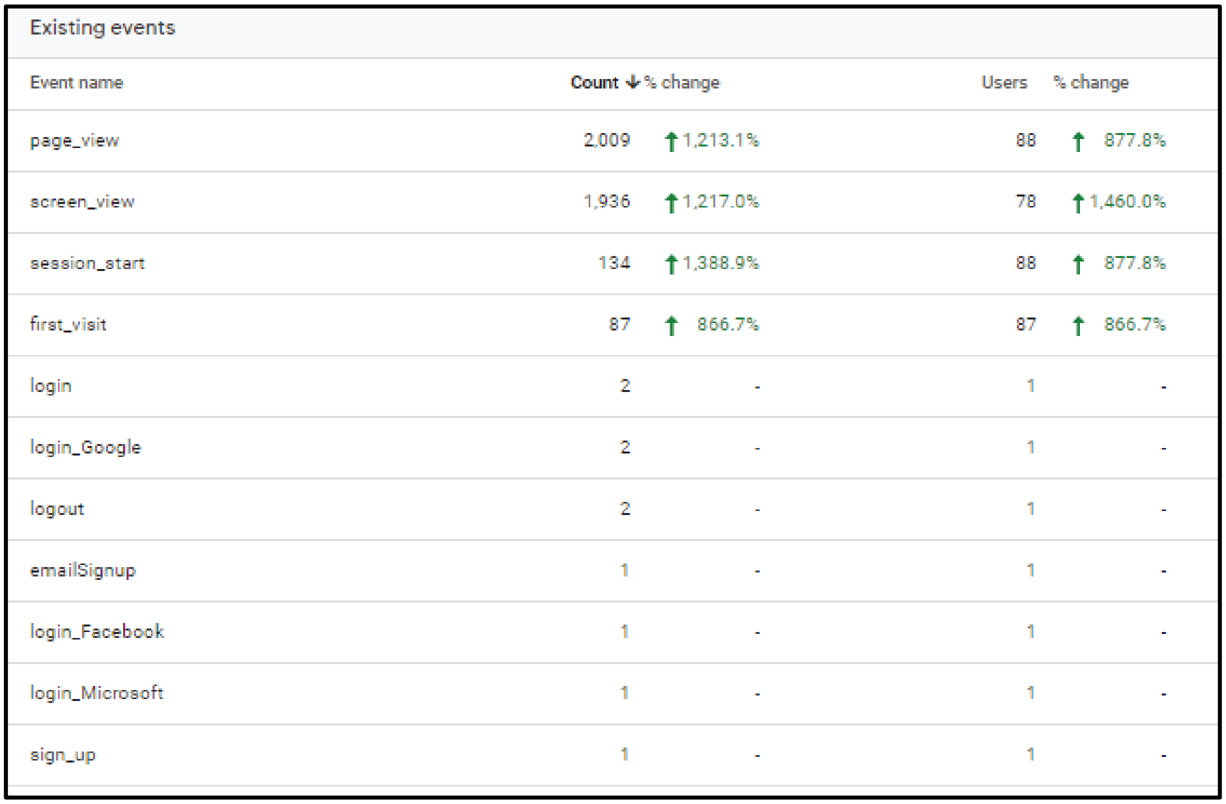


Figure 3: R-SRAFVP User Activity Diagram



	Country	Region	Users	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count All events
	Totals		88 100% of total	87 100% of total	117 100% of total	87.31% Avg 0%	1.33 Avg 0%	2m 43s Avg 0%	5,191 100% of total
1	United States	Michigan	73	72	103	88.79%	1.41	2m 26s	5,086
2	United States	Illinois	10	10	13	100%	1.30	6m 11s	81
3	United States	Iowa	4	4	0	0%	0.00	0m 00s	14
4	United States	Pennsylvania	1	1	1	100%	1.00	0m 07s	10

Figure 4: R-SRAFVP Usage Location Diagram



Existing events

Event name	Count	% change	Users	% change
page_view	2,009	↑ 1,213.1%	88	↑ 877.8%
screen_view	1,936	↑ 1,217.0%	78	↑ 1,460.0%
session_start	134	↑ 1,388.9%	88	↑ 877.8%
first_visit	87	↑ 866.7%	87	↑ 866.7%
login	2	-	1	-
login_Google	2	-	1	-
logout	2	-	1	-
emailSignup	1	-	1	-
login_Facebook	1	-	1	-
login_Microsoft	1	-	1	-
sign_up	1	-	1	-

Figure 5: R-SRAFVP Events Diagram

Conclusions

Although it may be argued that using software engineering discipline and deliverables seem like a waste of time, by implementing them on this project, time and effort were reduced over not using them. Although eleven deliverables were created, the outcomes showed during each phase change where there was little to no rework for requirements or design. Software engineering, even for small projects, can reduce effort by avoiding rework.

Coupling software engineering with cloud computing concepts continued to reduce development effort. Cloud computing helped to eliminate having a dedicated native application for each target platform. It provided pre-built services in the cloud and ability to have a cloud-based database as a service. The success of cloud computing can be seen through the two releases to production and operational metrics showing usage. With the R-SRAFVP application in use by the public, this is a good success for using both. In conclusion, using software engineering principles and cloud computing concepts would help any small project to be realized.

Acknowledgements

- Dr. Mary Warren, PhD, OTR/L, SCLV, FAOTA, for granting permission to use the R-SRAFVP for this project and serving as a primary reviewer for the requirements and design for the new web application.
- Dr. Beth Barstow, PhD, OTR/L, SCLV, FAOTA, for granting permission to use the R-SRAFVP for this project on behalf of the University of Alabama Birmingham and serving as a primary reviewer for the requirements and design for the new web application.
- Christina Hedlich, MS, OTR/L, SCLV for serving as a primary reviewer for the requirements and design for the new web application and field evaluating the application as it became public.

Appendix

All artifacts for this project, including source code, can be found on Github at:

- https://github.com/hedlichk/rsrafvp-app

The released version of this project can be experience on:

- https://www.rsrafvp.com

Bibliography

“Introduction to Progressive Web Apps - Progressive Web Apps (PWAs): MDN.” *Progressive Web Apps (PWAs)* | *MDN*, https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Introduction.

“R-SRAFVP Toolkit.” www.uab.edu/shp/ot/post-professional/low-vision-gc/student-resources. University of Alabama at Birmingham, 21 Feb. 2018. ZIP.

“Software Engineering: Incremental Process Model.” *GeeksforGeeks*, 5 Dec. 2019, https://www.geeksforgeeks.org/software-engineering-incremental-process-model.

“The HIPAA Privacy Rule.” *HHS.gov*, US Department of Health and Human Services, 13 July 2021, https://www.hhs.gov/hipaa/for-professionals/privacy/index.html.

“The HIPAA Privacy Rule.” *HHS.gov*, US Department of Health and Human Services, 13 July 2021, https://www.hhs.gov/hipaa/for-professionals/security/index.html.

Mohiuddin, Omar. “R-SRAFVP App.” *Open as App - Instantly Open Your Data as an App*, https://www.openasapp.net/portal#!/client/app/ed9f4600-e824-4447-8d3f-5a550bf5c601.