Appendix 4.5.4
Special Considerations for the Development of Application and Software Prototypes

Software is becoming increasingly important in the healthcare industry. Good coding practices are essential when developing in health-oriented information technology (IT) solutions. The following information serves as a basic introduction to early-stage software development.

Software Design Methodology

Software is a broad term that refers to a set of instructions written for a central processing unit (CPU) to perform in order to complete a task. Software can be written using a variety of languages, and it exists in many different forms. For example, with a mobile health application, the app itself is a type of software. The mobile phone operating system is another type of software. The phone's firmware is also a software program that, among other things, controls the hardware to allow the phone to connect to cellular towers. In the broader medical device space, software can range from the firmware inside a medical diagnostic machine, to desktop applications to view medical images, to enterprise-level applications such as an electronic medical record (EMR) system. Cloud-based applications are also becoming more and more prevalent in health care. Cloud applications typically use a "software as a service" (SaaS) model, where the application is hosted on the Internet using a network of remote servers instead of being installed locally on a personal computer. For instance, cloud applications in healthcare are already being used to store protected health information (PHI) for applications, run algorithms on remote servers to process data, and transform disparate data from various sources into a common form for storage and analysis in a database.

Each type of software has its own unique development and testing challenges. For instance, debugging and testing firmware is substantially different than debugging a cloud-based application. Since firmware is closely tied to hardware, testing the firmware often requires its own dedicated test instruments such as a logic analyzer and oscilloscope. A cloud-based system, on the other hand, would need to be tested for system redundancy and resilience against failure under extreme user loads. Enterprise applications, which are different than both mobile and desktop applications, have their own unique considerations, including the need to seamlessly integrate with existing enterprise applications, have robust auditing capabilities, and provide access control.

Regardless of which type of software innovators are developing, they can use one of two common methodologies for software design: (1) the waterfall method, and (2) the agile method. In the waterfall design methodology, each step needs to be completed before the next step can begin. Innovators begin by analyzing the entire problem that needs to be solved and defining specifications for each step of the solution. Then, they enter the design phase, which focuses on how to approach implementing the solution. Key design questions range from which system architecture, programming language, and libraries to use, to the inputs and outputs of each module and their function. The next step is the actual coding, followed by testing and maintenance. In waterfall design, significant effort is devoted to defining the specifications for each module before any coding begins. The downside to this type of design methodology is that any downstream change to the specification can necessitate massive changes to each of the designs in the subsequent steps. However, the waterfall design method does create robust documentation throughout the process, which is perhaps one of its strongest benefits and can be invaluable if the software will be subject to FDA review (see 4.2 Regulatory Basics for more information about U.S. regulation of mobile health technologies).

In agile development, the project is broken down into several manageable pieces. The core pieces are implemented first, to create a bare-bone, but functional application early in the project. Then, new features are added to the application. Each piece of the project is then implemented, often in short sprints. A sprint starts by defining the requirements for a particular block, implementing and testing that block, and seeking feedback from users. Feedback can then be incorporated back into that that block by refining the block's requirements. This iterative design process allows for the project to adapt to changing requirements while at the same time ensuring that the end product is what the user actually expects. The agile design methodology allows for changing specifications, which might arise from uncertainty in the healthcare environment.

Test-driven design is one form of agile development in which the design specifications are defined as a series of test cases. The test cases outline what the function is expected to do. Initially, all tests fail, as no code has been written. As the programmer writes code to implement features, test cases start to pass one by one until all of the test cases perform as desired. By writing test cases first, emphasis is placed on how one function should interface with others. By defining the interface for the function first, it ensures that the code written by multiple programmers will work seamlessly together. In addition, the tests serve as a type of technical documentation for the code as each test is effectively a design specification for the functional code. Other programmers can review the test cases to better understand what each function does and how to use the function. As the project evolves in complexity, another benefit to test-driven development is that if new code breaks any existing features, the programmer will be notified via failing tests. With all code written having a test case associated with it, innovators can ensure that the final product will conform to its specifications.

Selecting which methodology to follow depends on what type of software the team is writing. For example, firmware specifications are often well defined upfront from the hardware datasheet, so the waterfall method is generally a good method to follow. However, the agile method might be more suitable when writing a consumer application where specifications constantly change based on user feedback.

Prototyping Applications

If the innovators decide that the solution to their clinical need is an application, they will want to begin the development process with prototyping. Prototyping is critical because there is much more that goes into writing an application than just coding. Because it is relatively easy and inexpensive to begin coding, one common pitfall of application development is to start writing software before user needs are fully understood. This can lead to technological solutions looking for problems to solve, rather than needs-based innovation.

To avoid this risk, innovators should create a vision for the solution based on the need criteria outlined in the need specification. In doing so, they should keep in mind the context where the application will be used. For example, software designed for a desktop computer is different from software designed for mobile use. It is similarly important to fully understand the current user workflow and how the solution can seamlessly integrate into it. Any perceived inconvenience to the user could hinder the adoption of the solution.

Storyboarding

Storyboarding is an effective technique to explain the proposed solution to the target user and other relevant stakeholders. Typically drawn using paper and pencil, the storyboard allows the innovators to tell a story about how the application will be used to address the need. The storyboard is typically divided into multiple scenes, with each scene illustrating how a different element of the application will be used and in what context (details such as the graphical user interface are left to be detailed in later stages). With the storyboard, feedback can be solicited from key stakeholders (e.g., physicians or patients) to ensure that the proposed solution and workflow is positively received before coding begins.

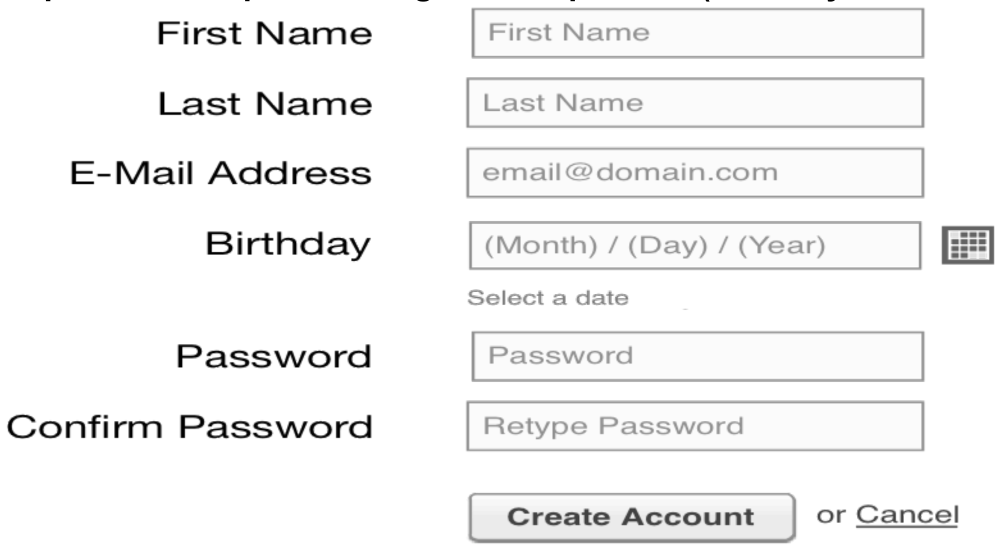
Graphical User Interface

After storyboarding, innovators can begin thinking about the design of the graphical user interface. Users interact with the application through its graphical user interface, and a thoughtfully designed interface goes a long way in creating a good perception of the solution. This is especially true if the application handles PHI. Because users cannot see the backend efforts the developer have made to securely handle PHI, they will often judge the security of the program on the professionalism of the graphical user interface. If the developers do not present a polished interface, the users may have doubts about the overall credibility of the system.

A good user interface does not solely depend on pretty graphics (although that can influence user perception.) The key factor to a well-designed application interface how easy and intuitive the workflow is to the user—it should be simple and self-explanatory enough that users can navigate the application without instructions. Getting to this point usually requires a significant investment of time, as well as multiple iterations. As with other forms of prototyping, innovators are encouraged to gather user feedback early and often.

When designing a use interface, it is essential to follow normal interface conventions and avoid adding overly complicate features to the system. For example, a submit button on a web page should look like an actual button, and not a text link or an icon. Additionally, it should be placed in a location a user would expect to find it, such as the bottom right of the page. In addition to affecting the user experience, innovators should recognize that the complexity of the workflow has an economic impact on the product since the more education/training that is needed, the higher the cost to use the system. One metric to measure simplicity is the number of clicks required to perform a certain task. If users can get the task done in two clicks versus three, the two-click method is almost always more desirable.

Figure 4.5.4-1 – The simplified user registration form on the right provides a better design for multiple reasons. The labels and the text input fields are separated, allowing the eye to easily follow the flow. All of the fields that require user input are aligned, including the submit button. The date input field displays a hint, showing the expected format. The submit button is clearly defined and labeled "Create Account," while the cancel button is now a link that guides the user to the next step. This differentiation eliminates confusion regarding which button to press to complete the registration process (courtesy of Frank Wang).

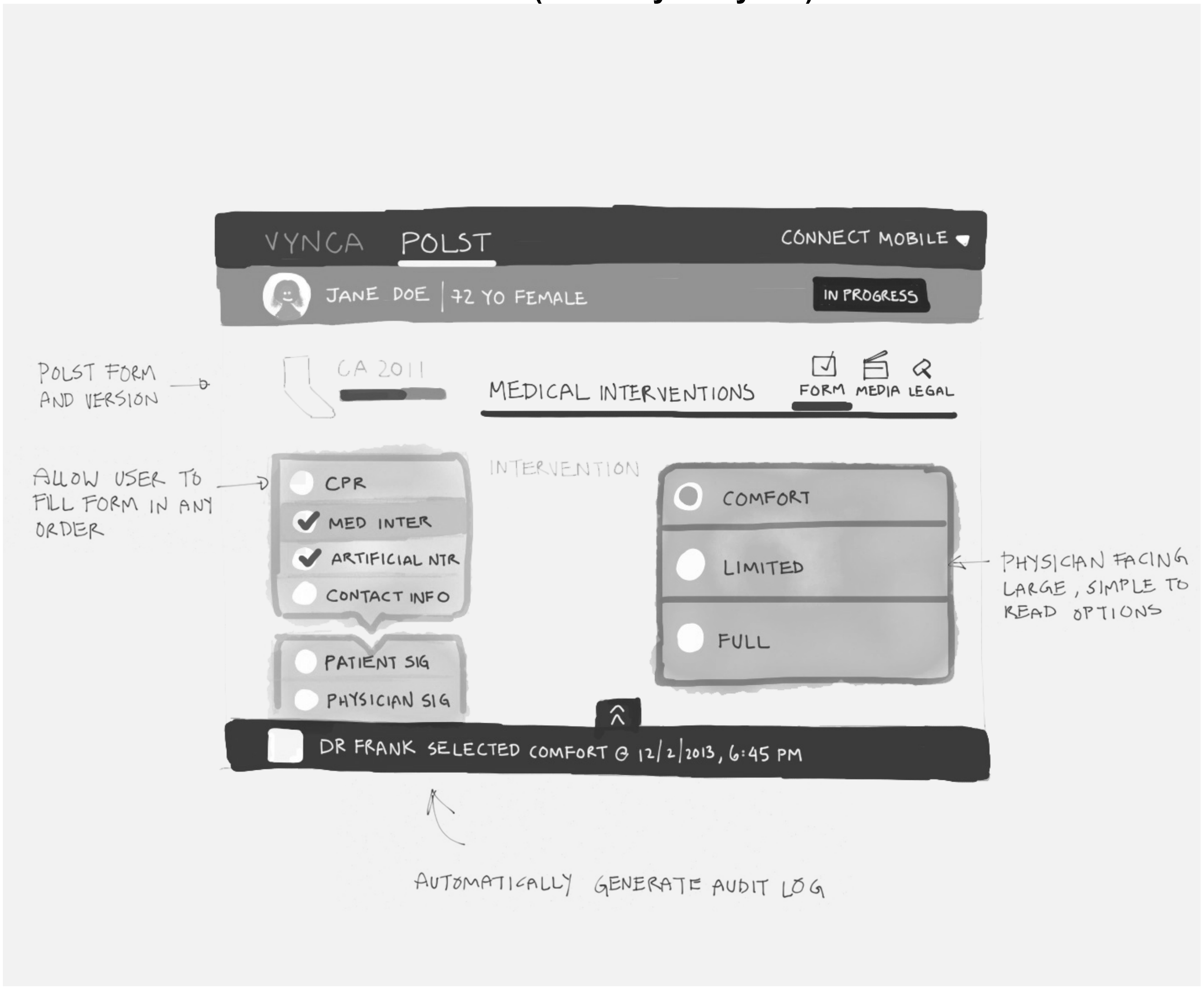


First Name	First Name
Last Name	Last Name
E-Mail Address	email@domain.com
Birthday	(Month) / (Day) / (Year) Select a date
Password	Password
Confirm Password	Retype Password
	Create Account or Cancel

One method for prototyping a graphical interface is to use wireframe sketches. Wireframe sketches are simplistic representations of how the application would look and feel, with the user elements represented using blocks. The wireframe sketches have just enough detail that users can imagine how the application will work, but not so much that they find it hard to give critical feedback. If the users show any confusion about how a task would be performed in the mockup, the innovators should try to understand where the confusion arises and investigate if it can be alleviated through a better interface design.

Once satisfied with the wireframe sketches, innovators can access templates in Photoshop, Illustrator, or PowerPoint to create pixel-perfect renderings of the images. Such templates typically contain many images of common user interface elements, allowing innovators to drag and drop the individual elements on the page to form the interface.

Figure 4.5.4-2 – A sample wireframe sketch for a health-related website includes just enough detail to explain how the page will function. It can be used to collect feedback from various stakeholders (courtesy of Vynca).



System Architecture and HIPAA Considerations

Once detailed user specifications are defined, the programmer(s) can start designing the system architecture to support the desired features. The system architecture can range from a simple native application to a combination of a native application with a remote database server to store PHI data in the cloud. In designing the architecture, it is important for the programmer to understand any relevant limitations, which might stem from constraints of hospital IT systems, HIPAA[1] regulations, or some other user-defined boundary conditions.

To determine limitations from a hospital IT perspective, it is beneficial to work closely with representatives of the hospital IT department. During storyboarding, hospital IT should be asked to validate the workflow and feasibility of the solution in a hospital-specific context. During system architecture design, these should also be involved in working out the implementation details for the solution. For example, if the solution depends on interfacing with the hospital EMR system, the programmers will need to determine what integration hooks are available that are compatible with hospital IT security policies. Or, if they are designing a website, they must determine which browser a particular hospital uses. This is important, for instance, since it can help determine if the website can utilize certain new HTML features that only modern browsers support. Working closely with representatives from hospital IT, programmers can create a detailed system architecture plan that will work within the hospital environment.

If the application deals with PHI, it will most likely be regulated under HIPAA. Many HIPAA requirements affect system architecture design, so innovators can save significant time and effort by designing these features into the application in the beginning, rather than as an afterthought. It is highly advisable to retain HIPAA counsel or a consultant to ensure that the application is fully HIPAA compliant.

The HIPAA privacy rule governs how companies can use and disclose PHI. For example, under HIPAA, any application must document when PHI is disclosed to other parties. The HIPAA security rule governs how the system secures PHI. For example, good design practice is to encrypt all data during transmission and rest. By encrypting data during transmission from the server to the user, it prevents a malicious third party from intercepting the transmission.[2] Encrypting the data at rest minimizes the risk of PHI disclosure if a malicious third party steals the server's hard drive server or a backup copy of the database. The security rule also requires the application to create an audit trail of how PHI is accessed within the system. If the application is disclosing PHI, it needs to verify that the people requesting such access are authorized to do so. From a development perspective, this requires some forethought since much of the information an application can use to authenticate a user is publically accessible online.

If the application requires the storage of PHI on a remote data server, the team will need to be sure to contract with a HIPAA compliant hosting provider. Innovators can work with that hosting provider to determine the optimal server architecture, data backup and recovery plan, and redundancy strategy to minimize downtime due to unforeseen events.

Coding

Though the actual coding of software applications is beyond the scope of this section, innovators will benefit from keeping a few tips in mind. Depending on the system architecture, whoever is selected to be the programmers for a project will need to decide on the tools most suitable for the coding task. This includes selecting the programming language(s) and the database to employ for the storage of the data. The programmers will also decide which software design methodology to use (e.g., the waterfall or the agile design method).

It is critical to have a means through which programmers can communicate with each other and with other members of the team. There are various ways to do this, including the use of a revision control repository, which creates a record of edits made to the source code (many such open-source tools are available online). A bug or feature tracker is another important tool for communication. These trackers allow non-programmers to participate in the coding process, by submitting requests for new features or by reporting bugs. They also allow everyone to track the progress of development. Documenting the source code is also crucial to facilitate communication between programmers. Although not strictly required for HIPAA compliance, it is good design practice (again, many open-source tools are available online).

Conclusion

Based on this overview, innovators should be able to prototype the initial feel of an application and then communicate the vision to the programmers who will be responsible for implementing it. Ideally, programmers with familiarity with healthcare software and its caveats will be involved from early in the project. Regardless, all innovators should remember to understand relevant regulations so they can design the solution with these constraints in mind from the start. Additionally, they are encouraged to continually seek feedback from users throughout the design and development process to ensure that the application does, indeed, solve the key user problems without introducing new ones.

Additional Resources

- Jonathan Anderson, John McRee, Robb Wilson, *Effective UI: The Art of Building Great User Experience in Software* (O'Reilly Media, 2010) – A good introduction to project management for software design.
- Bill Scott, Theresa Neil, *Designing Web Interfaces: Principles and Patterns for Rich Interactions* (O'Reilly Media, 2009) – Important insights on developing consistent, effective user interfaces for the web.
- Jonathan Rasmussion, *The Agile Samurai: How Agile Masters Deliver Great Software* (Pragmatic Bookshelf, 2010) – And easy-to-read introduction to various aspects of agile software development.

[1] HIPAA stands for the Health Insurance Portability and Accountability Act, which is the federal law that protects personal medical information and recognizes the rights to relevant medical information of family caregivers and others directly involved in providing or paying for care.
[2] A simple way to encrypt data during transmission is to use Transportation Layer Security (TLS) by purchasing a Secure Sockets Layer (SSL) certificate. This encryption method is employed when a user visits a site that begins with HTTPS, such as a banking or credit card website. Securing data at rest will require a bit more thought.