Team G

Realistic Issues

**Commercialization Issues (Linh Nhan)**

The first step in commercializing our Personal Health Assistant, PHA, for short, mobile application is to determine how we plan on earning revenue, if any, from the product as well as how we plan to market the product. One of the important aspects of commercializing a new product is to advertise and advocate for it so that people will want to use the application. PHA has two main audiences, the patients and the providers, that it must appeal and cater to in order for it to become successful. If either of these two parties does not participate in using PHA, then the application will fail as a product. In order to be able to partner with healthcare providers, we would need to form a small company that consists of developers that will be able to maintain and update the product when necessary. Since we cannot predict how successful the product will be, we will not form a large company where we will have much more to lose if the product fails. “Many high-tech companies started large, poorly conceived, and poorly justified e-commerce initiatives during the dot-com boom, only to see their investments wasted during the bust” (Goyal 9). Once we have a stable footing with the product, we will be able to expand. In addition to having our engineers maintaining the product itself, we must have someone keep track of our financial progress. Our financial plan can be completed by financial advisors we hire into our small company as employees or we can hire a company to do it for us. If our budget is really tight and low, then we have the option of maintaining the financial records ourselves, but it would be optimal to have someone who has the proper education and training do it for us.

 The reason why we would want to form a small company as opposed to approaching another company is because we want to have ownership over the code we write for the application. It will also be easier to have the number of people accessing the patient data restricted so that the security and legal issues revolved around working with an application that contains sensitive information will be constrained to a known group. Additionally, allowing the same people to maintain and update the code as the product progresses will be important because the people writing the code will understand it more closely. Furthermore, we cannot make the code open source for people to update since we have HIPAA considerations to keep in mind when working with the code. Because we will be working with actual data for customers such as their medical history, we cannot make that data open to the public. Our application will be working closely with our server and database where sensitive information is kept. Since we cannot release these things to the general public, we must have a defined group of individuals or employees that are under a contract and cannot expose the data legally. When developing and improving the application, we have ownership considerations to discuss with the developers as well as the providers we work with to implement more functionality into PHA. The person who created the idea of using a mobile application to assist providers in providing care to their patients will own the ownership to that idea. Any substantial improvement idea built on top of that can be patented or copyrighted can be granted to the individual who thought of the idea. Finally, the developers working on the application will have ownership over the code they write and contribute.

In order to draw users to the application, we can make it initially free for them to download and use. At the same time, however, we will not make all of the features available in the free application or we can set constraints on the use of the application. In these two ways, we would give the consumer a working, although not with all of the features possible and available in the full version, application that allows them to experience whether or not they will be continuing to use the product. For example, we can allow the user, in the free application, to interact with one doctor or provider. If they want to have interactions and care from more providers through the use of the application, they can pay a small fee such as $0.99 or $1.99 for each additional provider. Some of the features that we can make available with the purchase of the full application are picture upload for medications and setting up reminders for taking tests or appointments. Retaining some of the features of PHA for the full paid version of the application, which will be priced around $1.99, is an option that can be employed if more revenue is required for the success of PHA. Another way to earn revenue from PHA is to seek assistance from companies that want to advertise their medications. For example, companies such as Claritin-D or Zoloft could use PHA to have an advertisement between actions the user performs. Since PHA is closely related to medications and the individuals using it are tending to some issue they have, advertising will appeal to companies who want to reach a wider audience. Seeing advertisements relevant to the needs of a user will assist in the promotion of their product. Furthermore, we can offer an advertisement free version of the application for an upgrade fee of $0.99. In this way, users, who would rather skip the ads, will have the ability to and the ones who do not will benefit the companies who are paying us to advertise their product.

PHA development will be strongly based on customer feedback since we need to develop based on customer needs, which are constantly changing. We will have incremental phases of the product that will be tested, but will not be released to the public. We will have a test group that we will collaborate with to determine what features should be added or removed from the releases we give them to test. “Customer needs are always changing. It is critical for a company to be able to respond fast to changes in demand” (Goyal 5). By having incremental releases that we allow groups to test, we will be able to make changes as we continue with development. Because we have a strong need for customer reviews and suggestions when developing PHA, we will be working closely with both providers and patients to see what they view as important in an application such as PHA. “Customers are taking on a greater role in product development. This is because customers want increasingly customized products to fit their needs and more options tuned to their particular industry and company” (Goyal 6). For example, a cardiologist and an asthma specialist may require different things to be recorded in the patient’s PHA; we will need to customize the tabs for each of these to cater to the needs of the provider. We have features that we are planning to implement such as the graphing history as well as medication information for PHA, but we will discuss what features the patient and provider will find useful and remove the ones that are not. In essence, we will be implementing an agile software development method in order to launch the final product successfully. The man hours required for developing PHA will be dependent on how substantial the changes the providers and patients find and suggest. Because the needs of customers are constantly changing, we are not able to provide a rough estimate for hours at this point in time. However, we will have a small team of developers maintaining the application after it is launched in the event of software issues that arise as new technologies are released. We will have to design PHA to be able to adapt to changes as they occur in the dynamic field of computing.

**Works Cited**

Goyal, Jay. “Commercializing New Technology Profitably and Quickly”. Oracle. 2006. http://www.oracle.com/lad/industries/high-tech/022564.pdf. Web.

**Security Issues (Jose R.)**

The applications require users to enter their personal health care information to be able to interact with MSHV. This creates various privacy and security issues for which there are regulations to follow, namely HIPAA (Health Insurance Portability and Accountability Act). This act gives a guideline for which every application dealing with sensitive data must abide by rules to protect its users. The core of the applications is the use of MSHV, which us already a very secure system that gives tools to its developers in order to keep the standard. However steps still need to be taken once the Health information leaves the grasp of MSHV and resides in the memory of the applications or middle layer server.

HIPPA has many subtopics, but the two most prominent ones are Privacy rules and Security rules. To summarize the security rule, it protects all health information a covered entity creates, receives, maintains or transmits in electronic form. The general rules include:

1. Ensure the confidentiality, integrity, and availability of all e-PHI (electronic protected health information) they create, receive, maintain or transmit;
2. Identify and protect against reasonably anticipated threats to the security or integrity of the information;
3. Protect against reasonably anticipated, impermissible uses or disclosures; and
4. Ensure compliance by their workforce.

The Security Rule defines “confidentiality” to mean that e-PHI is not available or disclosed to unauthorized persons. The Security Rule's confidentiality requirements support the Privacy Rule's prohibitions against improper uses and disclosures of PHI. The Security rule also promotes the two additional goals of maintaining the integrity and availability of e-PHI. Under the Security Rule, “integrity” means that e-PHI is not altered or destroyed in an unauthorized manner. “Availability” means that e-PHI is accessible and usable on demand by an authorized person.

 To ensure that all these rules are followed, the applications must realistically and carefully follow more technical safeguards listed below:

Access Control. A covered entity must implement technical policies and procedures that allow only authorized persons to access electronic protected health information. Both applications will require a secure login by the server and HV before any data can be accessed. MSHV will securely hold any information unless the user is successfully authenticated by the use of RecordIDs and UserIDs. The connection to HV can be used in the context of only one UserID at a time, but you can change the UserID on the connection object between calls to pull data.

**Audit Controls.** A covered entity must implement hardware, software, and/or procedural mechanisms to record and examine access and other activity in information systems that contain or use e-PHI. This is a realistic issue that must be implemented on the server. Transactions and information changes should be logged with detailed information and stored in the mysql database.

**Integrity Controls.** A covered entity must implement policies and procedures to ensure that e-PHI is not improperly altered or destroyed. Electronic measures must be put in place to confirm that e-PHI has not been improperly altered or destroyed. Server APIs that deal with MSHV and patient-provider mappings (RESTful) must be carefully constructed and tested to prevent data alteration without the correct authorization. This is still a work in progress as the amount of information needed to access is still expanding. Access to these API URLs must also be protected from security risks such as sql injection since they are web accessible from the server.

**Transmission Security.** A covered entity must implement technical security measures that guard against unauthorized access to e-PHI that is being transmitted over an electronic network. Calls to and from the server and encoded using JSON with the correct authentication “appIDs” that HV provides and with the use of digital certificates. Each transaction must check for the validity of the certificate, which can be easily done with HV’s API.

The data first lands on the middle layer server when getting a call from the mobile application. The server is nothing more than an “offline” access application connected to Health Vault. The difference being that the user is prompted to sign in once, and the application is then permitted to access the user’s data without a future sign-in. The server application has to first request offline authorization from its users explaining the access level that it will be requesting.

 Similar to the HIPAA security rules, the Privacy rule is to assure that individuals’ health information is properly protected while allowing the flow of information. A disclosure must be given when a request for access is presented. Any information must stay private unless permitted by the user to do otherwise. Users with the patient application will give access to the provider applications only if they have explicitly search for and added that provider. The user can also take away those rights in much the same way.

**Bibliography**

<http://www.hhs.gov/ocr/privacy/hipaa/understanding/summary/index.html> (HIPAA: Privacy Rule)

<http://www.hhs.gov/ocr/privacy/hipaa/understanding/srsummary.html> (HIPAA: Security Rule)

**Software Licensing (Che Chu)**

The Personal Health Assistant contains programs and development tools made by third party companies. Both commercial and open source software packages require a commercial or an open-source license if our application is using them. Violation of software license can lead to a serious lawsuit. Some precedent cases even result in multi-million fine. Although our violation may be unintentional and minor, we must scrutinize the license agreements in order to avoid this situation. In the following paragraphs, we are going to present our usage of software packages and software’s license and regulation.

MySQL is used in the Middle Layer Server for Personal Health Assistant’s Account System. We currently use MySQL Community Edition which is the open source version of MySQL. The Enterprise Edition comes with additional technical support and backup tools but we don’t need edition until we need to scale up PHA. MySQL is under the GNU General Public License (GPL license for short) with Free and Open Source Software License Exception (FOSS License Exception). GPL grants the end-user the following rights: 1. Execute the program for any purpose 2. Change the software to suit the needs 3. Share the changes you make. FOSS requires the user to obey GPL and not to include any work licensed under GPL if the user intends to distribute a modified version of MySQL. In our case, MySQL is an infrastructural and internal part of the Middle Layer Server, which means MySQL is only installed and executed on our server. Our patients and health providers won’t receive any part MySQL in their PHA. Therefore, our usage is not violating both licenses.

We chose Microsoft HealthVault as our online health information platform; therefore, our uses of HealthVault are under the regulation of Microsoft. Microsoft specifies that services owned by Microsoft Corporation (not including components owned by third parties who have granted Microsoft permission to use) can be licensed if we are granted explicit permission under the License Terms. The License Terms has four basic requirements that we must follow: 1. When referencing to a Microsoft service, use the full name of the product 2. Include the statement: “Used with permission from Microsoft.” 3. Our use may not be obscene or erotic 4. We may link to Microsoft content by text link or a clickable Microsoft product logo link. No other images may be used as a link to a Microsoft site.

The Personal Health Assistant is developed using Titanium Studio as a development platform, which uses JavaScript as the programming language. Titanium Studio is owned and developed by Appcelerator Incorporate. It is under a free software license called Apache License 2.0 that allows the user of the software the freedom to use the software to distribute it, to modify it. Appcelerator offers different service plans: App Explore (Community Edition) and App Accelerate (Enterprise Edition). We are currently enrolled in the App Explore plan which is free of charge. In “App Explore License Agreement for Appcelerator Titanium Studio”, Appcelerator specifies that users are only granted with the right to install and use Titanium Studio solely to develop applications. The user must agree certain restrictions stated in the legal agreement, such as (a) not to modify, translate, or create derivative works from Studio (b) not to reverse engineer, decompile Studio, without the permission from Appcelerator (c) Studio may not be copied, etc. The above statements show that Appcelerator Studio is commercial open-source software which contains both proprietary and open-source components. Nevertheless, since we are not going to modify Studio itself but use it as a tool, we are not going encounter any legal problem. In addition, the free-of-charge App Explore program grants ownership of application to developers and user content contained in the application but Titanium Studio still belongs to Appcelerator Inc. under all circumstance. But like other open source software, we have to reference to it in our application if we use it as a part of our development.

**References:**

1. Commercial License for OEMs, ISVs, and Vars, MySQL Q&A<http://www.mysql.com/about/legal/licensing/oem/>

2. GNU General Public License:<http://www.gnu.org/licenses/gpl.html>

3. FOSS License Exception:<http://www.mysql.com/about/legal/licensing/foss-exception/>

4. Use of Microsoft Copyrighted Content:<http://www.microsoft.com/en-us/legal/intellectualproperty/Permissions/default.aspx>

5. Titanium App Explore License Agreement:<http://www.appcelerator.com/legal/appexplore-agreement/>

 **Developing Across Platforms (Brittany DePoi)**

The goal of this project is to bring personal medical management to the patient and bridge the gap between patient and provider regardless of the type of mobile device. Before the development of cross platform tools, it was necessary for app developers to choose a platform or create and maintain code for multiple platform so as to not alienate target customers. Appcelerator Titanium is the tool chosen for this project to develop code that can be deployed cross platform. Thus, eliminating the need to maintain multiple pieces of code. However, it is important to understand the realistic issues that may accompany using this choice over native code.

Titanium is a pure javascript solution to the cross platform development movement. A JavaScript API uses native code to access device specific features. Meanwhile the applications JavaScript is interpreted using a JavaScript interpreter: JavaScriptCore for iOS and Mozilla Rhino for Android. It does not use a webview for deploying applications, like Apache Cordova (PhoneGap) and therefore does not have access to the same JavaScript frameworks such as jQuerryMobile.

As a result, the code developed for this project will have certain limitations. First, because it is not strictly a cross compiler (in the sense that the JavaScript that is written is not translated into Objective-C and then compiled, it creates a symbol mapping that is translated at runtime) performance is a consideration. Commercial grade performance is another realistic issue that this will affect. The translation of the Titanium code provides close to native performance, but because it is not completely native code it will have worse performance than native code in specific instances. This is something to consider throughout the prototyping and commercialization process to make the application as fast as possible. Also to consider, is that although Titanium developers have worked to have Titanium code almost indistinguishable from the same direct object-c code, Titanium has been shown to perform better on iOS than Android devices. This will effect the testing and the layout of the code to ensure acceptable performance on all devices. It might also limit the scope of what we are able to accomplish in the given time frame because all of the code needs to be altered slightly after being tested on one device to work on another.

Also important to consider is the reliance that using Titanium to develop cross platform places on the maintainability of the code. Though it is possible to incorporate new native code by creating a Titanium module, the ability of the code to keep up with the technology will be dependent on the Titanium API as well as the JavaScript compilers of the devices. It is very important once the customer downloads the application on his or her mobile device, that the application will continue to work even if the operating system is upgraded. If Apple decides to upgrade their operating system, Titanium needs to repond to get a release of Titanium out that will work on the operating system. This dependency on Appcelerator to maintain their JavaScript API might create a lag in the functionality of the application for users, or require the user to wait until this functionality is addressed to update the operating system if s/he wants to ensure the use of the app. Furthermore, when a new operating system for a mobile device is released and it contains new native features, the code for this project is reliant on Titanium to provide the new API functionality before the new features can be utilized.

Due to the versatility of platforms we wish to bring to this project, understanding the tool that we are utilizing vital. Realistic issues for the project in terms of cross platform development are shaped by Titamium’s strengths, weaknesses, performance issues, dependencies and maintainability. Testing must be extensive and continue to be extensive with each update of any of the mobile operating systems or the Titanium development platform.

**References:**

1. Appcelerator Developers

<http://www.appcelerator.com/developers/>

**Ethical Issues (Wei Cheng Lin):**

Working on a system which involves using, transmitting, and editing personal health records can present both ethical and legal issues. Many aspects of the PHA System use personal health records to give feedback/suggestion to the patient or provider. This creates both ethical and legal issues in how the system uses the patient information and how the data can be used to change the way the user live in their daily lives. There are several components that have to be addressed in the design and implementation of the system in order to create an ethical system for the end user and to minimize legal issues that may appear in the future.

 Autonomy is one of the major components when creating and designing an ethical health care system like the PHA. By definition, autonomy in the context means letting “individuals make their own choices and develop their own lives in the context of a particular society and in dialogue with that society.” This means that any ethical health care system such as the PHA must allow the patient to have complete control of their health. When designing a system such as the PHA, autonomy must be preserved. However, one of the main functionality of the PHA system is to give recommendation based on health records given by the patient. For example, the system is likely to suggest the patient to consume certain foods in order to increase their intake of certain vitamins and/or minerals.

Another part patient autonomy is embedded in electronic health records systems (EHRs). In the modern world, increasingly, people are putting their health records online to a HER system. This allows the people to have easy access to their health records but it also creates ethical and privacy issues in how the data is kept and who can access the data. The PHA system uses Microsoft HealthVault as the EHR as the source of healthcare data. However, the PHA system itself does not have control of Microsoft HealthVault so therefore, the ethical issues from Microsoft HealthVault will transfer to the PHA application. As a part of the PHA system, a middle layer server is setup for extra security and privacy features such as setting permissions on who can access and/or edit their account. Protecting user data is a critical part of ensuring an ethical system.

Another important part in designing an ethical system is giving accurate information. As a health providing system such as the PHA, a bug in the code of the system can potentially be dangerous or even deadly for the user. The application has to have sufficient testing before it is released for commercialization. However, even with a robust system, an error in calculation can still occur if given incorrect data by the user. Therefore, it is important for the system to do self-testing to make sure all the data input and output is within reasonable range. For example, the system should detect an error when wrong units are given and alert the user. However smart the system might be, there is still no replacing a physical doctor/provider to monitor the patient. The user has to be made aware of the fact that the PHA system cannot and will not replace having a real physician in terms of providing health care.

There are many ways a system can be made to be “more ethical” but the fact is that there are no foolproof security and failsafe systems out there. However, there are many things that can improve the quality of healthcare and the PHA system is only a tool in which patients and providers can use to do this. Since electronic health record system is fairly a new technology, the ethical standards are rapidly changing and the PHA system must be built with change in mind. The maintainability and scalability is also key components that will make this system reflect of what is ethical.

**Works Cited:**

I. D. Norman M. K. Aikins, and F. N. Binka. “Ethics and Electronic Health Information Technology: Challenges for Evidence-Based Medicine and the Physician-Patient Relationship”. Ghana Medical Association 2011. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3266146/

**Intellectual Property Issues: (M. Swircenski)**

 Intellectual property laws are a significant presence in software development. Many pieces of software depend on intellectual property rights to function, and others have failed because of some intellectual property right they have violated. Thus, intellectual property laws have to be taken into careful consideration when doing the PHA project.

 The intellectual property rights to the project will be held jointly between all six of us. This will likely be until we graduate, after which they will be probably passed to whomever (if anyone) works on the project after us.

 We most likely not try to patent any discoveries we make or any processes we invent. Obtaining a patent costs upwards of a thousand dollars, and we are not operating under that sort of budget, nor do I expect that we will make that sort of money from this project.

 Infringement on existing patents is very probable. Thousands of software patents are filed each year, and it would take a highly trained team of patent lawyers to make sure that PHA is not infringing upon any of them. A cursory search of the online USPTO (United States Patent and Trademark Office) Patent Full-Text and Image Database of the terms “Medical information” and “online database” turns up a few patents that look like they might apply to our program. Most of these patents, such as 8532287, Method for proving a user with a service for accessing and collecting personal health record, belong to MyMedicalRecords.com, Inc. MyMedicalRecords.com is a website that “enables individuals and families to safely maintain their medical records and other important documents, such as birth certificates, passports, insurance policies and wills, in one central location and instantly access them anytime from anywhere in the world using the Internet.” This is similar to the service provided by MS HealthVault that we are using, so any infringing material should fall on them, not us.

The PHA application currently connects to HealthVault via an external server. This server is located on the UConn campus, and thus is a university resource. This server is managed by Alberto, so we don’t have direct access to it. If the PHA apps were to go to market, we would need a more permanent solution that wouldn’t depend on the university’s resources.