# CSE 293 Final Report Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td>3.2</td>
</tr>
<tr>
<td></td>
<td>3.3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td>4.2</td>
</tr>
<tr>
<td></td>
<td>4.3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td>3.2</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td>4.2</td>
</tr>
<tr>
<td></td>
<td>4.3</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>5.1</td>
</tr>
<tr>
<td></td>
<td>5.2</td>
</tr>
<tr>
<td></td>
<td>5.3</td>
</tr>
<tr>
<td></td>
<td>5.4</td>
</tr>
</tbody>
</table>
1 General Overview

1.1 Introduction

Our assignment was to create a Smartphone application that connected with Google Health and focused on Observations of Daily Living (ODLs). For our teams application we decided to use Android platform. The name of our application is The Personal health Assistant (PHA).

The Personal Health Assistant is a tool which enables users to record information about their daily well being, and makes that information available through collective reports, all from their handheld Smartphone device. The application will run on Android 2.1, an open source, fully customizable mobile platform. PHA will use a central server to generate reports, and pull information from Google Health.

1.2 Goal

The goal of the project is to make easier a doctor’s job of diagnosing someone’s condition, while making the day to day health management of the patient easier as well. This application should reduce the number of required doctor visits. We wanted to give the patient a fun and easy interface that they will want to use.

1.3 Overview

The application will work as a set of modules, functioning together to achieve the proposed goals. The layout will tabbed, with four tabs along the top of each section, for each major module. The modules are Home, Wellness Diary, Medication Alarm, and Reports. The Login screen is the starting point of the application, and allows the user to log in with their Google credentials, and launch into the Home screen. The Wellness Diary gathers information about the user’s
immediate health status, letting the user rate their ODL status on a scale of 1 to 10. This data will be time-stamped, and uploaded to the server, generating a timeline of the patient’s wellness. The idea is to enable the patient to ensure their improved health over time. The Medication Alarm lets the user grab their medications from Google Health. Alarms can be set for the user to turn on and off. Once an alarm goes off it will notify the user even if the application is closed. The Reports module will put to use data gathered in the Wellness Diary, and generate graphical reports, showing the user a graph of their health over a selected period of time.

2 Server Overview

2.1 Introduction

The Java Server stands as the connection between the Android application and the services it needs to be connected to in order to provide the desired functionality put forth above in the client general overview. The server is programmed in the Java programming language, and thus is able to be executed and run from a wide array of computing systems. This gives the administrator of the server the control needed to secure the server and provide data and connection integrity.

2.2 Goal

The goal of implementing a back end server was to provide a layer of abstraction from the Android client and the services it connects to. This allows the client to on an independent maintenance and upgrade path than the group of services it connects to. The Android application does not need to be aware of the way in which the methods it calls upon for information are implemented, as long as they provide the expected data and in the expected form. Changes in functionality, implementation, or bugs local to any service will not affect the functionality or implementation of functionality on the applications end. Furthermore, if there is a bug or a problem with a service, for example Google Charts, a different implementation for generating charts could be used by the server, requiring no changes to the android application.
Another goal of the server was to provide an efficient and easy to implement interface for which to interface with Google Health, Google Charts, and the Database. Our team attempted to implement the Hibernate interface though repeated efforts resulted in the same outcome; Hibernate is not properly documented or supported by the developers. Not having to worry about the Hibernate interface allowed our group to focus on the data retrieved from Google Health and the Database, which was not overly difficult to implement into our applications.

2.3 Overview

The main class of the server is the JavaServer class. The main function of the JavaServer class is to provide the connection between the Android client and itself. This is done through the use of the XML-RPC libraries provided by the Apache foundation. The XML-RPC functionality provides an HTTP socket for which a client can connect and send serialized POJOs, “Plain Old Java Objects”. The Android client connects through HTTP to the Java Server and sends a string as well as a java data structure. The string allows the JavaServer class to recognize the desired method to be executed, with the method argument(s) of the data structure, which was sent by the client. Once a method is called upon through the use of the XML-RPC method, the server runs any logic or I/O and returns a serialized java object to the client.

The JavaServer class also houses the implementation of the JDBC database communication methods, which allow for communication to the database server. MySQL libraries provided by Oracle provide the methods java needs in order to communicate with the server.

Also part of the server is the GoogleHealth and ParseXML classes. The Google Health class implements java libraries provided by Google in order to authenticate to and retrieve XML profiles from Google Health, which contain all of a patients relevant medical data. The GoogleHealth class calls upon the ParseXML class, sending it a java XML object. The ParseXML class provides a way to extract the desired user information from the XML objects and returns a HashMap of the users information to the
GoogleHealth class. The GoogleHealth class then returns to the JavaServer class with any relevant information, which is then passed to the Android client.

The GoogleCharts class is also included in the Java Server. It serves as a way to encapsulate Wellness data retrieved from the database as a URL string sent to the Google Charts service. The Google Charts service returns a URL pointing to a hosted image of the chart or graph requested by the GoogleCharts class. This URL is returned to the JavaServer class and is then returned to Android client, which displays the image on the screen.

3 Major Changes

Overall, we have stuck to our initial plan with this assignment, with the exception of leaving out a few overambitious features. We noticed early on that some things would take too long to implement, and fell too far outside the original project goals. Those are documented as unimplemented features.

In some cases, we discovered an easier or better way to perform a task, resulting in a feature change.

3.1 Unimplemented Features

Alternative Notifications

Aside from local phone notifications, other forms of notifying the user were discussed. These include text messages, and emails.

Due to the fact that we could meet our goal in this area without implementing additional notification methods, we chose to keep it simple, and stick with phone status notifications only.
Wellness Notes

Along with reporting a value for each ODL, we initially proposed to add an optional Notes section, allowing the user add text, and even images, to each ODL submission. Special features for the section would include a speech-to-text module, which would transcribe an audible note into a textual one.

These features were too far outside of the original project goals, and would require a considerable amount of additional work to implement. They were dropped early in the development process.

Sending Data to Google Health

From the beginning, we hoped for the application to function as a secondary interface for some of the more important items in Google Health, reading and writing data stored in the user’s Google Health profile.

Once communication with Google Health was established, our team instead chose to focus on handling that data, and preparing it for use within the application. Modifying and sending data back to Google Health would have easily doubled the amount of work involved with the project. Instead, we focused on one-way communication, which helped scale back a number of features, and allow use to spend more times with the core ones.

Forwarding Reports

Once a report is requested, it is generated, and sent to the user. A forwarding feature would have enabled the user to send their report to someone else, or themselves, via email. Similar to the alternative notifications, this feature was not an essential one, and was left out for the sake of time and simplicity.
3.2 Feature Changes

Reports Date Range
The user has the choice to generate reports based on a time range. In Prototype 1, we thought to implement that feature with a slider, having the user choose only how far back to aggregate their data for the report. This did not allow the user the ability to change the end date for the time span. For example, the user could choose a time like 2 weeks, and the program would grab data from the last two weeks.

In our final prototype, we chose to let the user pick both a start and end date. This was done using Android DatePickers, which tied in well to the look and feel of the application, as well as the usability of our application.

Primary Notifications
In Prototype 1, our medication alarm notification method was not very clearly stated, but by Prototype 2, we were using pop-up notifications to alert the user about their medication alarms. This was easy to implement, and helped us save time while working on other features. However, it was obtrusive, and did not allow the user to ignore an alarm, without dismissing the notification.

This was changed in Prototype 3 to use Android’s native AlarmManager. The new approach provides a better user experience, and ties in nicely with Android’s user interface, so that notifications can be easily received while using other applications.

4 Team Assessment

4.1 What did you/your team accomplish in this course this semester?
James:
Overall our project was very successful. Our assignment was to implement a Smartphone application that connects to Google Health. We have achieved this by
creating an Android application that connects to Google and pulls information from their account. Our application has the three sections that we have proposed from the beginning, wellness diary, alarms, and reports section. All of these sections are implemented and work the way they were proposed in the last prototype. Our application may not accommodate all of the features that we wanted to but the features it does have, they work the way we wanted them to. I focused on the GUI side of this project. It took a while to get around how Android GUI system works. After countless amounts of tutorials I really got comfortable with developing GUIs. I think by having each team member focus on a specific area we really a good idea. I would have liked to know more about the server but Kevin was the best in our team in that area so he worked on that part the most.

Ishmael:

Our team was able to develop a working cell phone application, using third-party APIs to authenticate the user, and gather data. The goal was to make available to the user a mobile form of the information that was already being gathered and stored in Google Health. We accomplished this through our Android application. Even more, we tracked ODL data over time, and presented that data to the user in a friendly, useful manner.

Kevin:

Our team was able to successfully create a working Smartphone application for a device that is still very much in its infancy. We were also able to develop a sever in java which connects to the android application and provides the desired path of communication between the phone and the services it depends on to provide functionality to the user. We were able to successfully complete the major aspects of the application that we have set forth at the beginning of the design project. Even though we were not able to include very bit of functionality which we have hoped, the end result was still a
very well functioning personal health assistant application which met most of its design
goals and was simple, intuitive and feature rich.

4.2 What have you/your team learned through your accomplishments?

James:

Being able to adapt to the Android platform in the beginning was a little scary. We have never created Smartphone applications before so we were not too sure what to expect. After reading many tutorials and trying things out we figured out how the general Android SDK works. Being able to adapt to new environments is very important in the field of computer science and we believe that we have done a good job adapting to Androids.

Ishmael:

We worked with a handful of different technologies, like Android, Google Charts, Google Health, XML-RPC, and MySQL. I spent most of my time working on the Android application, creating and managing alarms, building XML layouts for the interface, and storing data in Android’s SQLite database. I also worked on dynamically generating the reports based on data from the server’s MySQL database. Most of the technical things I learned were about Android. We had to learn the basics to start anything in it, and we had to learn even more to do some of the things the application required.

Overall, I learned the importance of planning, and working together with you team mates. We got a lot more done when we programmed together, than when we did so on our own. I learned that planning should be taken very seriously, to avoid presumably small tasks taking a lot longer than planned, or features needing to be added at the last minute, to support unplanned parts. I also learned that starting small, and building working prototypes throughout the development process is important. It allows you to see what works and what doesn’t, and what features may be too difficult to implement. Having a working application was more
important than it being packed with features, so starting small helps you get to your first working prototype quicker.

**Kevin:**

The project taught me many things about software engineering and designing a working project. The most important thing that I learned was the importance of proper planning and research on the outset of a new project. My struggles in attempting to get the Hibernate interface to work taught me that I should research the different tools that are needed to implement the functionality that we have set forth in our design. I learned that sometimes using a particular tool that is meant to make implementation easier is not efficient if you cannot grasp the knowledge required to use the tool.

**4.3 What would you do differently?**

**James:**

One thing that we could have done differently is start off with a smaller scope. We started off thinking that we would have time to add lots of other features, things like camera support and speech to text support. That stuff never got done because we did not start off with the basics first. One of the big problems was not knowing how we were going to pull the data off of Google Heath.

**Ishmael:**

If I had the opportunity to do another team project of this nature, I would plan more thoroughly, asking more detailed questions, and not making any assumptions. Many times throughout the planning stages we glossed over details which seemed simple, but were discovered not to be later on. Addressing these issues earlier would have helped us build a better structured application, with more flexibility for future development.

Also, when work gets separated near the beginning of a project, it’s important to know what you will receive from your other team members, and be prepared for it. Separating work is a good thing, but that work should be well defined, without gaps.
Overall, we did a lot of things well in this project, considering the fact that we started
the semester with absolutely no knowledge of Android, and ended with a good looking
application, which functions jointly with a Java Server, and a third-party service (Google
Health).

Kevin:

One thing that I would have done differently is to better research the different
tools I had to implement in the project to analyze their viability in our project. I would
spent less time trying to get hibernate to work and more time refining and implementing
the java server. If I would have managed my time better I would have easily been able to
implement more Google Health functionality such as posting information in addition to
retrieving it. I might have also had the time to parse the XML profiles in a more efficient
way in order to increase the speed of communication and the range of data retrievable
from the XML profiles.

5 Possible Future Extensions

5.1 Security

Adding encryption to login information would help improve the security of our
application. That is currently sent in clear text, which is not safe to do. Encrypting
retrieved data from a Google Health profile would also be beneficial, considering the
sensitivity of a person’s health information. SSL or an HTTPS connection can be used in
order to encrypt the data transmitted between Google Health and our server.
Furthermore, the data that is shown on the console screen of the server should be hashed
or the information not be as descriptive about the usernames, passwords, and health data
being retrieved and queried from the database.

The connection between the java server and the database should also be encrypted
to protect the information being queried and returned to the java server. As well as secure
connection, extra user security on the database server would also be a good extension to
security. The connection between our server and Google Charts should not be on the list
of possible future extensions because personally identifiable information is not transmitted to an from the service.

5.2 Doctor Information

Adding a section for doctor information would help the user have convenient access to their doctor’s contact information, as well as the ability to submit reports directly to them. Doctor information would be included on a separate tab that would naturally also include the specifications of each type of doctor as well as their contact information. The doctor information screen would ideally have a list of all the doctors, their field of practice, contact information, and GUI hooks to allow one button presses to call or email the specific doctor.

5.3 Doctor Maps

A GUI button could also be included with the doctor information that would link / open Android’s Google Maps application. This would provide the user the address of all their doctors or a specific doctor. With the built in hooks to the Google Maps android application, the user would then have the ability to get directions to their doctor, see the closest doctor to them, and possibly see a 3D map of the doctor’s office. The map of the user’s doctors could also have the ability to show other doctors who accept the users insurance carrier; this functionality could be added to the PHA application by parsing the insurance provider of the user from Google Health. The insurance provider can then be searched in Google for a list of all the doctor’s offices that accept that form of insurance, thus finding their addresses to send to Google Maps.

5.4 MySQL Database Migration

One possible future change to the database would be to export it onto a hosted website into a MySQL database. This would allow the java server to access the database as well as any administrator to access the database regardless of the state of the workstation, local network security settings, or VPN availability. This would also allow
for faster queries to the database as well as added web functionality in the form of a website an administrator could use to monitor or administer the database.