Mobile Surveying Systems for Company Data Aggregation and Interpretation

By
Dimuth Kulasinghe,
in Unison with Team #4943 of Engineering Senior Design Capstone Project

A Thesis Submitted to the Honors College
In Partial Fulfillment of the Bachelors degree
With Honors

THE UNIVERSITY OF ARIZONA

MAY 2012

Approved by:

Dr. Ara Arabyan
Associate Professor and Associate Head of Undergraduate Studies
The University of Arizona Electronic Theses and Dissertations
Reproduction and Distribution Rights Form

| Name (Last, First, Middle)                        | Kulasinghe, Dimuth, Madushan          |
| Degree title (e.g. BA, BS, BSE, BSB, BFA):        | BS                                    |
| Honors area (e.g. Molecular and Cellular Biology, English, Studio Art): | Engineering                           |
| Date thesis submitted to Honors College:          | 5/2/12                                |
| Title of Honors thesis:                           | Mobile Surveying Systems for Company Data Aggregation and Interpretation |

I hereby grant to the University of Arizona Library the nonexclusive worldwide right to reproduce and distribute my dissertation or thesis and abstract (herein, the "licensed materials"), in whole or in part, in any and all media of distribution and in any format in existence now or developed in the future. I represent and warrant to the University of Arizona that the licensed materials are my original work, that I am the sole owner of all rights in and to the licensed materials, and that none of the licensed materials infringe or violate the rights of others. I further represent that I have obtained all necessary rights to permit the University of Arizona Library to reproduce and distribute any nonpublic third party software necessary to access, display, run or print my dissertation or thesis. I acknowledge that University of Arizona Library may elect not to distribute my dissertation or thesis in digital format if, in its reasonable judgment, it believes all such rights have not been secured.

Signed: [Signature]
Date: 5/2/12

Last updated: Nov 15, 2009
Final Project Report
Portable Survey Tool Project
ENGR 498B

May 2, 2012

Team # 4943

Roxan Cruz
Natasha Dennison
Rae Gargione
Dimuth Kulasinghe
Leonardo Montoya
Luke Seavitt
# Table of Contents

## Contents

0. Abstract ................................................................................................................................. 5
1. Introduction .......................................................................................................................... 5
2. System Requirements .......................................................................................................... 6
3. Summary of Previous Reports ............................................................................................. 8
4. Top-level Design of Final Design Concept ......................................................................... 8
5. Subsystems, Algorithms, Interfaces ..................................................................................... 9
6. Design Analysis ................................................................................................................... 17
7. Implementation / How it Works .......................................................................................... 20
8. Development Plan and Project Management ....................................................................... 32
9. Budget and Suppliers .......................................................................................................... 32
10. Acceptance Test Plan ......................................................................................................... 33
11. Conclusion ........................................................................................................................ 35

Appendix A: Glossary .............................................................................................................. 36
Appendix B: Project Diagrams ................................................................................................. 37
Appendix C: Pseudo Code ....................................................................................................... 39
Appendix D: Project Schedule ................................................................................................. 43
Appendix E: Individual Contributions .................................................................................... 46
Appendix F: System Requirements ......................................................................................... 48
Appendix G: Training Packages .............................................................................................. 52
Android Training Package ...................................................................................................... 52
Appendix H: Website ................................................................................................................ 80
Appendix I: Team Pictures ...................................................................................................... 82

---

Page 4 of 82
0. Abstract
The 4943 engineering senior design team at the University of Arizona developed the QuiPST mobile surveying system that allows for the quick construction and administration of surveys for use by clients. The project was sponsored by Raytheon, which deemed the software useful for collecting and interpreting data to improve the efficiency of the company. QuiPST is a client-server system that allows surveys to be responded to via Android and iPhone mobile platforms, as well as through web browsing. Surveys are created, and responses archived on a web server created through the Django web-framework. Users can manually trigger and respond to surveys, or be pinged randomly at requested time intervals. The resultant data collected and interpreted from the QuiPST system helped Raytheon in its evaluation of different project management styles.

1. Introduction
1.1 Scope of the Project
The Quick Portable Survey Tool (QuiPST) can be used by virtually anyone as a generic survey application. For the purposes of this project, QuiPST will be used as a time study application to investigate how specific Raytheon employees spend their time throughout the day. Wanting to collect this data in the least intrusive way, a randomly generated ‘ping’ will notify the users to take a survey, which asks them what they are currently working on. The data collected from the survey will be used by Raytheon to determine the efficiency of employees on the Critical Chain program. However, the QuiPST application itself is a client/server system that connects smart phones to the online server.

1.2 Problem Statement and Background Information
The original project was a Time Study application for RMS to investigate how employee time was utilized during a typical work day and to compare the efficiency of employees on the Critical Chain program as opposed to those who are not. A previous time study was done in the 1990s by Texas Instruments Missiles division (later acquired by Raytheon Missile Systems) that surveyed 65 engineers across 12 programs. A key find in the study was that engineers spent only a small fraction of their time at work designing. Since the study, Raytheon implemented several changes including increased automation, increased use of Critical Chain, and many environmental changes. The Critical Chain has had significant positive impact on many of Raytheon’s programs, however there are no internal or external studies that validates its impact on the value added time of individual employees. It could be argued that advances in technology have increased the number of interruptions and the impact of interruptions has not been qualified internally at Raytheon.

Although a previous study existed, Raytheon desired an updated study due to advances in technology. The use of smart phones to perform the study could greatly improve the results from the previous study as well as create an easy and accessible way to collect data from users.

Due to internal issues within Raytheon, the scope of the project was minimized to a generic survey application as opposed to an in depth time study. The application will still collect information from employees on the Critical Chain and how they manage their time on the clock, however only a small group of individuals on the Critical Chain will be participating so the comparison between those on the
Critical Chain and those who are not will not be made. This miniature survey will act as the pilot survey. If Raytheon chooses to perform a time study on a much larger scale, QuiPST can easily be implemented to collect more data.

1.3 Product Expectations
Before the survey has begun implementation, an empty database will be setup on the Virtual Private Server (VPS) that will collect answers from users. Questions will be stored in a survey tree and subsequent questions shall depend on the user’s previous answer. The survey tree will be stored on the server itself and the tree will be called at the time the user initializes a survey. The administrator has the ability to change questions in a tree or to add several trees on the server. Each question and answer will be mapped to the database and the server will collect answers correspondingly.

Upon first use of the application the user will be prompted to input a user identification number and password, which will be given by Raytheon after collecting demographic information separately. The user will then input his or her work schedule for the week under the settings tab of the application in order for the survey to be made available during the correct time. Scheduling can be changed as needed to account for any vacation time or change in schedules. The user may begin a survey at any time during the work day, so long as a survey is made available by the server. The time to complete the entire survey will be minimal, as to not take up too much time out of the employees normal daily activities. Along with the user having the option to manually start a survey, they will also be notified at random times throughout the day that they should take a survey, with the number of random alerts being set by the administrator.

2. System Requirements
The requirements for the QuiPST client/server system were identified by deriving information from the meetings the design team had with Raytheon. These requirements were categorized into six different segments and were considered to be the following:
- Functional Requirements
- Technology Requirements
- Performance Requirements
- Utilization Requirements
- Trade-Off Requirements
- Test Requirements

2.1 Functional Requirements
The functional requirements describe the central components and ideas that allow QuiPST to adhere to Raytheon’s product expectations.

- The system shall have a device application where users will input their data
- The system shall have a user setup
- The system shall allow administrator to setup certain characteristics of the application
- Administrator will have the ability to send out a trigger and a new question to all participants at random time intervals throughout the day
- The system shall have the application alert the user with notifications
- The system shall only recognize the inputs of the user by their User ID
- The system shall provide a training package for the user
• The system shall have a simplistic user interface design
• The system shall recognize a user not responding to an alert

2.2 Technology Requirements
The technology requirements describe the hardware and software components required to build and run QuiPST.

In the technology requirements, the design team had to identify what would be available and allowable for the QuiPST client/server model system. The high level proposed requirements under this category were considered to be the following:

• Client side requires a smart phone with either an Android version 2.x+ or iOS 3.x+ operating system with browser connectivity
• Client side requires the smart phone to be installed with the QuiPST application.
• The server side requires a local PC with the following installed:
  o Django framework version 1.3.1+ installed. See Appendix A for definition of Django.
  o Python version 2.7+ installed. See Appendix A for definition of Python.
• Imported QuiPST server-based project file
• Web connectivity

2.3 Performance Requirements
The functional requirements describe the relative performance standards expected from QuiPST in regards to time and efficiency.

• The client side should be able to notify users immediately upon receiving a survey request by the administrator
• The client side should not force the user to spend more than 30 seconds responding to any single survey.
• The server should be accessible by clients at all times.

2.4 Utilization Requirements
The utilization requirements defines how each technological component in QuiPST will be utilized.

• The client side will utilize the system services on each phone platform allowing for user notifications. See Appendix A for a definition of notifications.
• The client side will utilize the native components of each phone platform.
• The server side will utilize certain services and modules available to the user from the Django framework:
  o The server side will utilize basic database management scripts.
  o The server side will utilize and configure http request methods.
  o The server side will utilize database-syncing and server-running scripts.

2.5 Trade-off Requirements
The trade-off requirements introduce aspects of QuiPST that lead to various design tradeoffs.

• The client side requires user authentication.
• The server side needs to be able to aggregate data.
2.6 Test Requirements
The test requirements define steps we are taking in testing QuiPST before sending it for production.

- The client and server sides shall consist of software debugging phases.
- The server side shall consist of an integration phase during which QuiPST will be run on a temporary developmental server.

3. Summary of Previous Reports
Three major design concepts were evaluated. These three design concepts correlated to three major components of our system: Server Side, Client Side and Data Partitioning. After analysis of all of these, weighing different measures for each component, Django was selected for the Server Side, JQuery Mobile with some native development was selected for the Client Side and demographics being inside the Raytheon firewall, with UserIDs outside the firewall, was selected for Data Partitioning.

Django is a Python framework that is used as a medium for creating a database, which automatically generates tables and automatically writes SQL statements. Django is a tool built so others can use it as a tool. JQuery Mobile is a HTML5 based user interface system that allows HTML code to look, to the user, like a native application. This only can go so far, and then some native development will have to be done for both the Android and iPhones.

These decisions set up the implementation of QuiPST and left the rest up to the specific coding and analysis of these tools.

4. Top-level Design of Final Design Concept
QuiPST uses a client-server model for surveying and data aggregation. The client side is composed of the smart phones given to users of QuiPST, as well as the software components allowing the viewing of surveys and configuration of different settings. The server side allows for the creation and administration of surveys, data aggregation and archiving, and the running of business intelligence procedures on the resultant data. There is a constant cross-communication done between the client and server sides. The server side uploads different surveys and pushes them to the clients in the system. The clients then administer the surveys to the users; the resultant information is then aggregated at the server.

4.2 High Level Model
The Surveyer has two major parts associated with it: the Client Side and the Server Side. See Appendix B for flow chart.

4.3 High Level Server Side Model
Within the Server Side, a Project is created, called QuiPST and then within that project an Application was created, called Survey. See Appendix B for flow chart.

4.4 High Level Client Side Model
The Client side is composed of two major subcomponents: jQuery Mobile and native development. See Appendix B for flow chart.
5. Subsystems, Algorithms, Interfaces

The “QuiPST” survey application system is composed of two primary subsystems; Server and Client. The two subsystems are developed independently and composed of multiple modules. Each module is vital in order for the survey system to function properly.

5.1 Server

The “QuiPST” Server subsystem acts as the database for the survey application. This is where the questions for the application are pulled and sent to the Client. It is also where the data that is collected is stored. Administrators are the only users who have access to the Server to ensure security and efficiency.

Django, a high level Python web framework, was chosen to construct the relational database. Django acts as a medium for creating a database. It automatically generates tables and SQL statements once the Models, URLs, Views, and templates have been created. It comes with an object-relational mapper built in. Within this, the database layout can be described using Python programming language.

5.1.1 Models

A model is the single source of data about the data. The essential fields and behaviors of the data stored are contained within a model. Each model is a Python class with attributes that represent a database field. Through this Django provides automatically generated database access API. This allows the admin to create, retrieve, update and delete objects. See Appendix C for complete pseudo-code.

5.1.1.1 Group Class

The models.py file created for the “QuiPST” database is composed of several classes. These classes define the different categories associated with the project. For instance a “Group” class was created to fulfill the requirement that multiple surveys could be pulled by the Application from the database. Within this class is a field called “name”. This field allows the admin to assign a name to that particular Group. Each field created is specified as a class attribute, and these attributes map to a database column. The “Group” class acts as the primary key for the relational database. In other words it is the class that encompasses all the other ones.

```python
class Group(models.Model):
    name = models.CharField(max_length=128)
```

5.1.1.2 UserID Class

Once the “Group” class is created there needs to be a way to determine what user is part of that group. Thus a “UserID” class was constructed. Within this class two fields must be built; “number”, “group”. The “number” field allows a number to be assigned to a user so the admin knows which user is in which group. Because this is a many-to-one relationship, a positional argument is required to show the class to which the model is related. Through the use of a foreign key the “group” field shows the association between the “UserID” and “Group” class. The “UserID” class can now relate back to the “Group” class.

```python
class UserID(models.Model):
    number=models.CharField(max_length=400)
    group=models.ForeignKey(Group)
```

5.1.1.3 Question Class

Now that the “UserID” class has been constructed, the “Question” class can be developed. This class acts in almost identical fashion to the “UserID” class because “Question” relates back to “Group” via a Foreign Key statement. The only significant difference between is in the first field where you input the
questions you want to ask for that particular survey. Once those are created the “Group” class now can display which users are in the group as well as what questions they will be asked.

```python
class Question(models.Model):
    question = models.CharField(max_length=400)
    group = models.ForeignKey(\Group)
```

5.1.1.4 Answer Class
After the question has been created a set of answers must be given to answer the question. Thus an “Answer” class must be constructed. There are three fields associated with this class. Like the previous classes the “answer” field allows you to input the answers to display for the question. However, the second field is different. The “Question” class acts as the primary key for the “Answer” and “Response” classes. Where the two classes are foreign keys back to “Question” and “Question” is a foreign key to “Group”. Because the project implements a “tree of questions” where the next question to ask depends on the previous answer, another field must be added to the “Answer” class to make it complete. This is the “next_question” field. This field allows the admin to link the “answer” (with a foreign key) to the next desired question. Thus when a user gives a response to a question, the next question given is related to response.

```python
class Answer(models.Model):
    answer = models.CharField(max_length=256)
    question = models.ForeignKey(\Question)
    next_question = models.ForeignKey(\Question, blank=True, null=True, related_name="next_question_set")
```

5.1.1.5 Response Class
Finally, after the question has been displayed with answers to the user, a response can be submitted. This is done through the “Response” class. Here there are two different foreign keys. The first is within the field “user.” The “user” field relates back to the “UserID” class so that the user name can be displayed with the user identification number. The other is the “answer” field. This field relates back to the “Answer” class. It essentially is telling the “Answer” class what the user response was so that it can output the next question with another set of answers. This is how the “tree of questions” requirement is fulfilled. The “submitted” field is the last to be associated with the “Response” class. This field fulfills the time stamping requirement by outputting a date and time next to every user response.

```python
class Response(models.Model):
    user = models.ForeignKey(\UserID)
    answer = models.ForeignKey(\Answer)
    submitted = models.DateTimeField(auto_now=True)
```

5.1.1.6 Summary
To better understand the aforementioned relations it is easiest to look at the models.py file backwards. The user selects a response based upon the given answers. The answers are associated with a question that is given by an administrator. That question is given to a certain set of users who are identified with user ids. Those user ids are in turn related to a group whose name is determined by the administrator. The application can then trigger the database to pull that group which will allow the users of that group to answer the survey.

5.1.2 Views
A view function is a Python function that takes a Web request and returns a web response. This is done through HTML pages and 404 errors if there is an error. Within the Views folder, views.py, different functions are created that take in an input that is placed within parentheses. This input is “request” in almost all cases. Then something will be returned by this function. See Appendix C for pseudo-code.
5.1.2.1 Submit Answer Function
The submit_answer function has the functional use of allowing the user to submit an answer to the first survey question and then continue down that path of the survey until they are done, having the next question linking to the previous answer. This function defines an ‘answer’ and a ‘user’ which call upon the Answer and UserID classes respectively. They are getting specific objects from these classes. For the answers, an if/else loop allows for the survey.html page to display the next question while there are still linked answers, else it will return the done.html page and will complete that specific survey instance. See Appendix C for pseudo code.

5.1.2.2 CSV Function
The csv function has the functional use of extracting all the responses that have been collected for each user and exporting them to Excel. This is done through a CSV mime type. A request is made to the Responses class and then these responses are returned to the CSV.html page and then the data is able to be saved as a .csv file and opened in Excel. In addition, a .csv2 file was created that stores the pinging time and the response time in comparison. See Appendix C for pseudo-code of the CSV file.

```python
def csv(request):
    responses = Response.objects.all()
    return render_to_response('csv.html', {
        'responses': responses
    }, mimetype='text/csv')
```

5.1.2.3 Survey Function
The survey function has the functional use of getting the questions that are to be shown to the user taking the survey. This function defines a ‘user’ and a ‘question’ which will call upon the UserID and Question classes and extract the specific objects associated with those classes. This returns all information to survey.html. See Appendix C for pseudo-code of the Survey file.

```python
def survey(request):
    user = get_object_or_404(UserID, number=request.GET.get('user'))
    question = Question.objects.filter(next_question_set=None)[0]
    return render_to_response('survey.html', {
        'question': question,
        'user': user
    });
```

5.1.2.4 Index Function
The index function has the functional use of allowing only specific users to be able to have access to taking the survey. This function defines ‘users’ and gets these as objects from the UserID class. It returns all of these users to the index.html page. See Appendix C for pseudo-code of the Index file.

```python
def index(request):
    users = UserID.objects.all()
    return render_to_response('index.html', {
        'users': users
    });
```

5.1.3 URLs
The urls.py module acts as a simple mapping method between URL patterns and the created callback functions in the views.py file. When the client makes a request from the created Django site, it determines which URL to use. Django then loads the specified module and searches for the URL patterns (see URL patterns below).

Django will go through each pattern in order and stop at the first one that matches the request sent by the client. It then imports and calls on the created view of the matched URL found in the views.py
module. An HttpRequest is then passed as the view's first argument. Any values captured by in the regex act as remaining arguments. If no matches are made then an error view is displayed.

```python
from django.conf.urls.defaults import patterns, include,

urlpatterns = patterns('survey.views',
    url(r'^$', 'index', name='index'),
    url(r'^survey/$', 'survey', name='survey'),
    url(r'^csv/$', 'csv', name='csv'),
    url(r'^submit_answer/$', 'submit_answer', name='submit_answer'),
    url(r'^not/$', 'http_response', name='http_response'),
    url(r'^test/$', 'test', name='test')
)
```

### 5.1.4 Settings

The admin site is an automatically generated interface. This is done through the settings.py file located in the QuiPST project folder. This allows for us to enter administrators that will be the only ones who can log onto this page and access the setup. This is the main area of security in regards to who will be able to access the site and its associated information. This also is where the Linode server information would be entered and allow the site to “go live.” Under the installed_apps portion of Settings, our application, survey, is entered as to make sure that this is specifically loaded each time. The URLs mentioned above are called upon and thus linked to the /admin page.

### 5.2 Client

The QuiPST Client package is a phone application that allows users to answer surveys that were uploaded by the project administrator into the server. The client package is only necessary for the QuiPST system if the user desires being randomly pinged, as the actual surveys and calendar modifications are still performed through the web browser.

The client package is solely responsible for the administration of random pings. The user, after setting his calendar, will receive up to 10 pings throughout the day, at times specified by the user. The distribution of the pings is determined by the random pinging algorithm as specified in Section 6.4.

#### 5.2.1 jQuery Mobile Fundamentals

The client logic is written through a single HTML file; this is made possible by the utilization of the jQuery Mobile framework.

By importing the necessary cascading style sheets (CSS) and JavaScript libraries, jQuery Mobile will automatically alter the markup of any given DOM (Document Object Model) element in the HTML file so as to make it look and feel like an authentic smart phone application component. This is done through the data-role = "" parameter, telling the framework the type of markup that should be implemented. jQuery Mobile also allows the use of AJAX, allowing the user to write HTML scripts adhering to the framework’s design philosophy and allowing smoother page transitions.

#### 5.2.2 Client Package Design Fundamentals

The client side packages of Android and iPhone are both developed natively. The two packages differ in their architecture, but present similar experiences to the end user. Both packages run a random pinging algorithm as described in Section 6.4, and when chosen to ping, deliver a notification to the user, which
is immediately visible and gives a sound upon arrival. The user clicks on the notification and is immediately presented with the online survey.

5.2.2.1 Android Client Package Fundamentals
The Android QuiPST client package consists of three unique applications, being QuipstPro, QuipstBackground, and the third party event scheduling application Tasker. The package runs off the concept of a push-server, periodically checking to see whether a given time is appropriate for a survey. The Android package only relies on server communication for preliminary user authentication and calendar synchronization; afterwards it runs independently of the QuiPST server. This provides an advantage in system dependability, as the package does not require an internet connection to run. However, it also has a disadvantage in setup time, which takes significantly longer than the iPhone package. It also requires extra steps from the user’s perspective when making server-side changes, which must be synced with the client application. Finally, the periodic system checks run by Tasker may be interruptive to the user.

QuipstPro is the user-interactive component of the package. The user does not interact with the other applications except for setup. The user will enter his or her credentials (as provided through the QuiPST administrator), and after authentication, make changes to his or her calendar, and sync the calendar settings with the application.

QuipstBackground is the system-component of the package. It is a script running the actual random-ping algorithm, and decides when to notify the user to take a survey. It utilizes the calendar settings synced by the user in QuipstPro to run the algorithm; until this step is completed, QuipstBackground will not run.

Tasker is the scheduling component of the package. Its job is simply to independently run QuipstBackground every two minutes.

The three applications depend on one another to give the user the experience of random pings.

5.2.2.2 iPhone Client Package Fundamentals
The iPhone QuiPST client package consists of a single application, simply called Quipst. The iPhone package runs off the notion of a pull-server, with actual scheduling and decision making being done from the QuiPST server; the application acts as a remote listener which forwards (and notifies) users of signals being sent by the server. This provides an advantage in setup time and simplicity from the user’s standpoint, as the user installs the application and is immediately ready to go. It has a disadvantage in system dependability, however, due to the need for constant communication with the server to be able to receive alerts.

The user installs QuiPST, and enters credentials for user authentication (as provided by the QuiPST administrator). The QuiPST server interprets and verifies the request, afterwards creating a virtual phone entity for the specific user. The QuiPST server contains a python script that runs the random ping algorithm for all stored phone entities, this script is run every minute. When a given phone entity is chosen for being pinged, the server sends a push notification to the given phone, and the phone in turn notifies the user to take a survey.

5.2.3 QuiPST Design Components
QuiPST utilizes a comprehensive user interface that gives users full control over the application. It will
incorporate the following components:

5.2.3.1 Welcome Page
After the user enters their user name and password they are directed to the home or welcome page of the application. The welcome page of the client side simply provides a menu screen to the user, allowing them to access various parts of the application. This includes moving to the “Launch Calendar” button, “Synching” the calendar once it has been completed, Resetting the User ID, and Start Survey. Links between the buttons are constructed through HTML ref links, which can be easily converted into a button using the jQueryMobile “data-role” tag. See Appendix C for pseudo-code.

5.2.3.2 Calendar Page
The Calendar page allows the user to configure the times at which he or she wants the application to alert them. The user selects the desired day that they want to edit and they input the times that they will be working (excluding lunch hours). The operator also has the option of applying times for one day to the entire week if they work a set schedule. Then by selecting the save button their inputted schedule will be saved and they will be brought back to the home page so they can “Sync” their calendar with the database. The user will also have the option of choosing not to be alerted for an entire day by selecting the “Don’t Alert Today” button. It is important to note that the default time range settings are Monday through Friday, 9:00 AM to 5:00 PM thus if the user does not work those hours each week they must be deleted before the new times can be inputted. See Appendix C for pseudo-code.

5.2.3.3 Sync Calendar Button
The sync calendar button allows the user to take the inputted calendar information stored on the application and sync it with the server so the server knows when and when not to send a survey to that specific device.

5.2.3.4 Reset User ID Button
Due to a limited number of devices, multiple people must be able to use the same smart phone. The reset user id button allows for this to occur. Each user is assigned a User ID and Password that is specific
to them. Once data is no longer needed from a user the phone can be retrieved and the “Reset User ID” button can be selected deleting the information associated with that user and allowing for a new operator to input their User ID and Password as well as the calendar settings specific to their work routine.

5.2.3.5 Start Survey Page
QuiPST will alert the user of a new survey through the use of notifications. The user can choose to acknowledge the notification at any time; he or she will automatically be transported to the QuiPST main screen. The user can then select the “Start Survey” button to begin the survey.

The survey itself can be completed by answering questions linearly. A specific page is shown to the user for any question, which will present the question and a set of components allowing the user to answer that specific question. Answers will be in the form of single choice selections followed by a time spent performing task question using a slider bar. The user always has the ability to move back to a previous question should a mistake been made. Once finished with the survey, QuiPST will thank the user for their input, and send the resultant survey file to the server.

5.3 Trade Studies
While designing each sub-system trade studies were performed to support each decision. They were performed while constructing both the Server and Client sub-systems.

5.3.1 Server
A trade study was performed as to which database framework should be used to develop in within the Server sub-system.

5.3.1.1 Django
When first designing the database for the server sub-system of the Quipst project, several factors were considered. Due to a lack in database development experience amongst the team, functionality and user-friendliness were an important factor in determining which framework to develop in. It was also important that the chosen framework provided a strong support package that could be readily accessed. Two database developmental frameworks were found that supported the aforementioned desired factors. Both Django and Microsoft SQL could have been used to create the database sub-system for the server aspect of Quipst.

Upon further research Django clearly became the obvious choice. It provided an in depth detailed support package that includes sample python code and explanations about how to construct a database. Django also automatically generates the SQL statements that would otherwise need to written should Microsoft SQL be the framework of choice. Due to the simplicity and extensive support website that
Django provided it was an obvious choice to develop in python using Django as the framework rather than Microsoft SQL.

5.3.2 Client
A trade study was performed as to which frameworks should be used to develop in within the Client sub-system.

5.3.2.1 Native Development
The scope of the project called for the use of smart phones to administer the surveys and because the potential users of the application were thought to own either Android or iPhone smart phones, applications needed to be developed for both markets. The difficulty in this was the development environment for Android and iPhone are completely different - Android applications are built using Java using programs like Eclipse, while the iPhone is developed in objective-C using Xcode. While some team members are knowledgeable in Java and Eclipse and have basic knowledge of the C programming language, no members have had experience using objective-C and more importantly, no experience with Xcode. With a deadline of early February 2012 for the completed product, the team was unsure if learning a new language and programming the application was plausible.

Another option was to build a web application that could be accessed through the mobile phone’s web browser instead of using a native application. Using this option, the survey would be available to any phone with a web browser. The third option was to build a hybrid web and native application using the mobile framework PhoneGap. The PhoneGap framework is based on HTML5 and JavaScript which is programmed like a web application using HTML code but can be ported onto several different smart phone platforms.

While the PhoneGap framework seemed useful, it was deemed unnecessary in the end due to its lack of advantages over a simple web browser for the purposes of our project. The PhoneGap application simply read data from the web server and presented it in the jQueryMobile CSS format, but this could be achieved through a simple browser as well; PhoneGap was decided to be an unnecessary extra step in the development of the QuiPST system. It was decided that the client side applications would be designed for the exclusive purpose of administering the random pinging implementation; the actual survey taking did not require them at all. The random pinging implementation was decided to be producible for the Xcode development environment, due to the iPhones extensive covering of push notification technology as it was. The Android side was decided to be similarly feasible.

After consulting outside resources and much research, it was decided that using PhoneGap would be the most ideal. PhoneGap requires that the SDK be installed in the development environments and instead of coding different classes, the code would be done in different HTML files. Using PhoneGap still required that the application developers be knowledgeable in HTML, which some members have had experience with. Essentially with PhoneGap, the same HTML code used in writing the Android application could be used for the iPhone application and would look and function in the same manner. PhoneGap would still be able to use the phone’s native APIs, but make it easier to program the application and for these reasons, PhoneGap was the obvious choice.

5.3.2.2 jQuery Mobile
While researching the PhoneGap mobile framework, it was also suggested by outside resources that we also look into using the jQuery Mobile web framework for the user interface. Doing so allows the programmer to utilize the jQuery JavaScript Library and make use of different elements in the library
including buttons, tool bars, and layouts which made the HTML code feel like an actual native application to the user. Using the different elements can be done using tags that have the same syntax as HTML tags.

It was decided that jQuery Mobile would be used mainly for aesthetic reasons since it would be more appealing to the user and much easier to use. jQuery Mobile uses AJAX in page to page transitions, allowing the application to make calls to and from the server asynchronously without interfering with the user’s experience.

6. Design Analysis

6.1 System Integration Introduction
The full QuiPST product is achieved through the integration of the client and server subsystems; the two communicate with each other to collect and aggregate data on QuiPST users. Django traditionally works as a pull-server, but has the ability to perform limited push notification services due to the flexibility of the APN push notification cloud service as given by Apple. The same flexibility is not as apparent in the Android platforms’ C2DM push notification cloud service, however. Because of the easier implementation of push notification services, the iPhone client-server system utilized push technology, while the Android client-server system utilized pull technology. See Appendix A for a definition of pull-server.

6.2 Client Side Requests
All pull-services from the client side are run through HTTP (POST) and (GET) methods. See Appendix A for definitions.

6.2.1 HTTP (GET)
To receive information from the server side, the client side will send HTTP (GET) methods that include as parameters the server side functions which it is requesting responses from, the path for the method in its URL, and any extra parameters the server requires.

```javascript
$(function() {
    $.get("/survey/xhr_test", function(data) {
        alert(data);
    });
});
```

The above code snippet would request a response from the xhr_test method, and output that response to the user.

6.2.2 HTTP (POST)
HTTP (POST) methods will be used to input information into the server database. The parameters are similar to the HTTP(GET) method, requiring a URL that specifically defines the server side functions being utilized, as well as the parameters to be read and inserted by the server.

```javascript
$.post("/survey/xhr_test", {
    id: 123456,
    answer: "Yes"
},
    function(data) {
        alert(data);
    };
```
6.3 Server Side Responses
The server will then respond to POST and GET methods sent by the client. Django defines its responses through the HttpRequest and HttpResponse modules.

6.3.1 HttpRequest and HttpResponse
Clients contact the server requesting specific service methods created in QuiPST; the service method utilizes the HttpRequest module to determine the type of request (POST or GET), and returns information to the client as an HttpResponse object. The HttpRequest module services are preconfigured in the Django framework; QuiPST separately defines the HttpResponse object to be passed back to the client for each method.

```python
from django.http import HttpResponse

def xhr_test(request):
    if request.is_get():
        message = "Hello AJAX"
    else:
        message = "Hello"
    return HttpResponse(message)
```

In this code snippet, the xhr_test method responds to inputs by the client. The request parameter and is_ajax() method defining the request as a HTTP (GET) protocol are configured through the HttpRequest module. The method then returns an HttpResponse object with the desired data.

6.3.2 URL Path Environment Variables
URL destinations are written in a specific python script url.py as mention in section 5.1.3; the destinations use regular expressions and project file paths to direct client requests to their corresponding methods.

```python
from django.conf.urls import patterns, url, include

urlpatterns = patterns('',
    (r'^quipst/survey/$', survey.views.xhr_test),
)
```

r'^quipst/survey/$', survey.views.xhr_test’, for example, would lead a client request to the xhr_test method given above.

6.4 Pinging Algorithm
The QuiPST pinging notification system is made possible through a random pinging algorithm created by the QuiPST design team. The goals of the algorithm are to give the end user the experience of receiving the number of daily pings (quota) as issued by the administrator, and making the times at which the pings are received seemingly random, while adhering to the user’s calendar settings. The pinging algorithm implemented on the client phones and server, for the Android and iPhone respectively.

The algorithm checks the number of pings a given user has currently received, and subtracts this from the quota, giving the number of pings left for QuiPST to administer. A cool-down period is hardcoded by the administrator, which is a minimum time that must be passed before QuiPST can attempt to ping again. This cool-down time is multiplied by the number of pings left, giving the minimum number of minutes required by the QuiPST system to deliver the remaining pings to the user. The algorithm then
figures the number of minutes QuiPST has left in the day to work with, based on the current time, and calendar settings set for the current day. The required number of minutes is divided by the minutes left in the system, giving a ratio that is called the urgency of the system. There is a maximum urgency rating hardcoded by the administrator, and if the calculated urgency is above that, it is reduced to the maximum urgency. Finally, the percentage chance of this final urgency is the percentage chance that the user will be pinged at any given time. See Appendix C for pseudo code.

6.4.1 Android
The Android QuiPST client package has each individual phone running its own pinging algorithm under QB, as described in Section 6.4. The user stores data on current pings received and hardcoded elements such as cool-down period, quota, and max urgency in the phone’s internal storage, which QB then processes. Tasker runs the QB application once every two minutes. The pinging algorithm is written in Java with the integrated Android sdk.

6.4.2 iPhone
The iPhone client package does not run the pinging algorithm; the algorithm is run from the server side for every registered iPhone every minute. If a phone is to be pinged, a push notification is sent to the phone as described in Section 5.3.2.1. The pinging algorithm is written in Python using the Django web framework.

6.5 QuiPST Client-Server Communication Run through
As mentioned in Section 5.2.2, the client-server communication style differs vastly for the Android and iPhone client packages.

The Android system relies simply on HTTP (GET) functionality. When the user enters his or her credentials into the phone, QuipstPro forwards this information to the server; the server simply validates the information by responding with a Boolean reply. If the message request was successful, QuipstPro completes the rest of the application setup and readies the system for use.

The Android system also uses HTTP (GET) functionality to synchronize the user’s online calendar with the phone. QuipstPro requests metadata on the user’s personal calendar from the server, and once granted this information, stores it on the phone’s local storage. At this point, the Android client package runs independently of the server. Accessing of the user’s surveys and calendar are done through hyperlinks that simply open up traditional web browsers.

The iPhone system is more restrictive on push notifications than the Android. In order for iPhones to use Apple’s push notification service, the server and device must communicate with the Apple Push Notification Service (APNs). The device only needs to maintain one constant connection to the APNs. The structure of the APNs is such that if the server wants to contact an application installed on a device, the server contacts the APNs which then delivers a push notification to intended devices.

In order to use the APNs in a testing environment, a development SSL certificate request is generated. An Application ID is generated by the developer and the App ID is configured for push notifications using the certificate signing request. The certificate is then downloaded to the development environment and also uploaded to the server. A provisioning profile must be created to authenticate testing devices. Under this provisioning profile, devices are registered as testers for the application and the certificate is
used to verify devices. The server side code will call the sandbox APNs to send a message and include the certificate as verification. See Section 7.3.2 for more downloading information.

6.6 Communication Failures
Communication failures for the Android client package occur only during setup and during the taking of online surveys; there are no built in restraints against communication failures here because the system does not depend on constant communication with the server, and sporadic communication issues while taking surveys are largely dependent on the user’s wifi connection and do not damage the systems processes overall.

7. Implementation / How it Works
7.1 How QuiPST works
The QuiPST is comprised of QuiPSTPro, QB, QuiPST Backend, Tasker and a website.

7.1.2 QuiPSTPro
QuiPSTPro is the icon on the smart phone that, from the user’s perspective, is the actual application. A user training package on specific instructions on how to set up the phone and start using Quipst can be found in Appendix G.

However, QuiPSTPro is nothing more than a login page and then home screen that communicated both with QB- QuiPSTBackground and the Server.

Once login has been verified, the home screen will appear. This screen has four buttons. Take A Survey, Launch Calendar, Sync Calendar, Reset User ID.

Note: The user will only be asked to log in one time in the beginning of their use of QuiPSTPro. This is to allow for minimal extra steps during the survey time, and make it as easy on the users as possible, to not have to remember to log in every day.

7.1.2.1 Take A Survey Button
This will allow the user to manually take a survey at any given time. This button will pull up an internet page that will allow the user to take the survey.

7.1.2.2 Launch Calendar Button

This will allow the user to edit their calendar settings by linking them to an internet page. Upon this, there will be a message directing them back to the QuipstPro App to finish their calendar setup.

7.1.2.3 Sync Calendar Button

This button must be pressed to actually sync the calendar settings the user just set with the QB. The technicality of allows for the QB, the pinger, to know when to start and stop the pinging process. (More to come in the next section).

7.1.2.4 Reset User ID Button

When a user has finished their data collection time, they have the option to log out of QuipstPro and their information will be ‘turned off’ until they log back in at another time. This is mainly beneficial for testing when multiple users will be sharing the same phones during different testing groups, and it can be sure that the newest user is not still signed in as an old users ID number.

Once the user has set up their calendar and synced it, the user would never technically have to open the QuipstPro app again, as they will just be receiving random pings that tell them to take a survey and direct them to the internet page where the survey is found.

7.1.3 QB- QuipstBackground

From the users perspective, they will never see or know about QB, or QuipstBackground. That is because it is just an additional app running in the background. Its main function is to generate the random pings. QB will be run by Tasker every two minutes. (More information on Tasker in the next section). Each time it is run, QB will check the calendar settings and see if the user is in a time frame to be pinged. If they are, QB will then check its algorithm and determine when to ping the user. The pings will not happen within 20 minutes of each other and will be randomly generated throughout the day, with a total of 10 per day. If all criteria is met, then QB will send a notification to the user and will let them know they should take a survey. Clicking on the notification itself will bring up the survey, making it flawless to the user of what they should be doing at the time.

7.1.4 Tasker

Tasker is an outside app that has the main purpose of completing a task. The task in this case is to run QB every two minutes, and then QB will work according to the information above.

7.1.5 Testing

Three main stages of testing took place for QuiPST. After each stage, feedback was received and changes were made, to give the final version of QuiPST.

1. Alpha Testing: Personal Android and iPhones of our team members were used to test the application.
2. Beta Testing: ZTE Droid phones were purchased for Raytheon and deployed to a small group at Raytheon. User friendliness and anomalies were addressed in this stage.
3. QuiPSTPro Testing: Based off of specific functionality of the ZTE Droid, reworking of the code for QuiPST was necessary, specifically regarding a sleep function of the phone, and so QuiPSTPro was developed.

QuiPSTPro 2.5 became the final design of application after a month of testing and rework.

7.1.6 Website
An informational website, quipst.com, allows for detailed information on the capabilities of QuiPST for other potential users. The website also is where you could login from a computer and access your survey, if you are a user, or access the administrator page, where surveys can be altered or created, if you are an administrator. See Appendix H for visuals of how the website it set up.

7.2 Administrator Responsibilities
The administrator is responsible for entering the allowed user IDs and passwords, setting up the survey questions and answers, and then extracting the data.

7.2.1 Entering User IDs and passwords
To keep your survey secure, the Administrator will have to enter in all the User IDs and passwords that will be allowed to access the specific survey group.

This can be done by clicking the “add” associated with “User ids”

Now you can add the user number and password. The number area will only accept numbers, where the password area will accept both numbers and letters. Each User ID has to be associated with a group, which in this case, is Critical Chain. Since more than one user will be added, clicking “save and add another” will allow you to add multiple users without leaving the screen.

Once this is done, only users with the correct User ID and password will be able to access the survey from their mobile phones.

7.2.2 Setting up the Survey
When you want to create a new survey, the following steps should be taken.
First, you need to create a group. This can be accessed through quipst.com/admin/survey. This group will be specific for the type of survey you are creating.

Once you have clicked on “Add” to add a Group, you can name the group and click “Save”

You can see that your group “Critical Chain” has been saved, and that now you have two groups in your queue. Now you need to return to the quipst.com/admin/survey page, which can be done by clicking on “Survey” in the upper left corner under “Django administration”.

Now you will want to click on the “add” that corresponds with Question.

You can type in the first question you want your survey to ask, and then select the group you want the question to go in. This can be done by clicking the drop down menu. In this case, we want to add it to the group we just made called “Critical Chain.” Click save and then return to the
quipst.com/admin/survey again.

Now that you have a question, you want to assign all the possible answers to this question. This can be done by clicking “add” that corresponds to Answers.

Type one possible answer to this question. In this case, the answer is “Working on the CC task that was assigned to me”. Then, use the drop down arrow to associate this answer to the appropriate question.

Below this, there is the field called “Next question.” This is an optional field. If you want to associate another question with the answer you just typed in, you will click the green arrow next to this New Question field. A new window will pop up, and allow you to write a new question, and associate that question back to the group again.

In this case, if a user answered that they are “Working on the CC task assigned to me” then a follow up question would be “How long have you been doing what you are doing now?” Once again, the group that should be selected is Critical Chain. Click save.

This pop up window will close and you will be able to see that the “Next Question” field has your new question in it. Click save. You can see that your information has been saved.
Now if you return to quipst.com/admin/survey, you can continue the process above of adding more answers and questions to your survey “tree.”

For example, another answer to the question “What are you doing right now?” could be “In a meeting (not related to my CC task).” This can be done by clicking the “add” button associated with Answer. You can see that now in your drop down menu for “Next Question” you have now a list of the previous questions you have made and can reuse or create a new one.

Click save and repeat this as many times as you want. The survey has no maximum number of questions or answers. A visual representation of the survey created above is shown below.

Note: If you want to add an answer to a question that will allow for a slider bar of time values to be how the user will enter the time they have been working on a specific task, you will simply type in “SLIDER_1-50”, in all capitals, which will output a slider on the phone or web page.

7.2.3 Extracting the data
When your data collection time is over, you can extract it from the server by going to the URL http://backend.quipst.com/survey/csv. This link will immediately prompt you to ‘Save As” and click
“Okay.”

Once this downloads, the file type must be changed to .csv, which means comma separated value. This can be done by simply right clicking on the file name when it appears under “Downloads” and selecting “Open Containing Folder”. Once this is open, the file will have an ending of .part, but this needs to be changed to .csv.

Note: A warning message may appear when trying to change the file type, but ignore this, click “Yes”, and proceed.

Now the file can be opened in excel and all the data will be available for analysis, starting with UserID, Question, Answer, Ping Date/Time, Response Date/Time.
7.3 How to Download

7.3.1 Android
The entire QuiPST android client package can be downloaded from the QuiPST team’s personal Dropbox. The applications consist of apk files that must be downloaded to the phone, and can be installed through the use of a file browsing application. Third party application installation (outside of the Android market) must be allowed from the users phone to achieve this. The application may eventually be hosted on the Android market, but no immediate plans have been set.

7.3.1.1 Tasker Download
Tasker has to be downloaded separately, since it is an outside application. A free 7day trial can be used or the app can be purchased for about $5.

7.3.1.2 Tasker Setup
Once this is downloaded, Tasker will have to be set up with QB.

Initially, Tasker must be downloaded onto the phones. The Tasker home screen will be blank. A new task can be created by clicking the green plus sign in the bottom middle of the screen. Make sure that you are under the “Profiles” tab at the top of the screen when adding this new task. You will be asked to name the profile, but click the check to continue with no name.

Then, a screen will appear that allows you to pick from several different options. Select “Time”, the second option in the list.
Uncheck all boxes but the “Repeat” box. Set these parameters to be “every 2 minutes.” Select the green check in the bottom left corner to save changes.

Next you will be prompted to add a “New Task” to this Time profile you have created. Click the plus sign on this to add a new task. Once again, you will be prompted to name this task, but click the check to continue with no name.

The task will initially be blank, and you will have to click the plus sign in the bottom right corner to add a task. A new screen will pop up that displays a list of different task options. Select “Load App.”
Now, scroll to find the QB app that has appeared in a list of all your applications. Select it. Another screen will appear once you have selected QB. Click the green check mark in the lower left hand corner.

![Images of the QB app and the green check mark.]

Now, you will be brought to a screen summarizing the Task you have just created. Make sure that the screen looks like the one below and click the green check mark in the lower left hand corner. Then you will be brought to another screen summarizing the Profile you have just created. Make sure the screen looks like the one below and click the green check mark in the lower left hand corner.

![Images of the Task summary and Profile summary screens.]

Now click the green check mark once more to save your Task. You can now close Tasker and wait for your QuiPST notification.

### 7.3.2 IPhone

In order to download the QuiPST application onto an iPhone, the iPhone must be registered under the development license. Every iPhone has a Unique Device Identifier (UDID) that can be accessed through iTunes. With the iPhone connected to a computer, iTunes must be opened and the device selected on the left menu. A summary of the device will appear to the right including name, software version, etc. The UDID can be viewed by clicking “Serial Number” in the summary. This number should be copied and saved into the provisioning portal of the Apple developer license.
Up to 100 devices can be registered for development purposes. The developer must login to the iOS Development Center, select the iOS Provisioning Portal and Devices. A list of currently registered devices will appear and the developer will have the option to “Add Devices.” The device name and UDID should be entered here.

To allow the application to receive push notifications, on the development computer, launch the Keychain Access application on Mac OS X should be launched. Under the “KeyChain Access” menu, select “Certificate Assistant”, and then “Request a Certificate From a Certificate Authority”. It will then request the email address and a name for the certificate that should be saved to disk.

After the certificate is saved, an App ID must be created and configured for push notifications. In the iOS Provisioning Profile, select App ID from the left menu and a list of App IDs will be listed. To the right of
the list will be a configure button that should be clicked. There will be an option that says “Enable for Apple Push Notification service,” that should be checked and the Certificate Signing Request wizard will appear. Select continue to move on to the submission screen and choose the Certificate Signing Request that was saved earlier. Apple will then generate the SSL Certificate that must be downloaded and will be used for push notifications. Double click the downloaded certificate to install it in the Keychain Access application. This certificate will be used to contact the APNs to allow push notifications to be sent.

The next step is to create a provisioning profile. On the Provisioning Portal, select Provisioning from the left menu and click New Profile. Input a name for the profile, select the certificate and App ID previously created and all devices will be used for push notifications, then click submit. The provisioning profile will be pending approval, and after approved must be downloaded to the development computer. The profile will have a name corresponding to the one selected previously and will have syntax similar to “MyProfile.mobileprovision.”

To install the application on a device, connect it to the development computer. Drag the provisioning profile into Xcode, then launch the Organizer application from Xcode to select the currently connected device. Under “Provisioning,” the profile just added should be listed.

In Xcode, select the project name, then Build Settings to view properties of the application. Under “Code Signing Identity,” the provisioning profile should be selected. The device is now ready to load the application. In the dropdown menu next to the “Run” and “Stop” buttons, select iOS Device and click play to load the application. After “Build Succeeded” is displayed, the phone is ready to be disconnected and the application ready to receive push notifications. For the complete pinging algorithm, see Section 6.4.

### 7.4 Technical Documentation
All server login information, domain name information and other technical documentation, important for Raytheon, who will need to access these files in the future is given below.

**Linode Server:**
Login Info-
- Username: 498SrDesign, Password: team4943
Website- manager.linode.com
Payment- $19.99 per month and paid for till End of December

**Domain Name:**
-Quipst.com purchased from namecheap.com bought for $9.98 for 1 year

**Django Admin:**
Website- quipst.com/admin
Login Info-
- Username: Raytheon, Password: administrator1234

**Django Coding:**
- Python:
- Relational database

**Mobile Coding:**
- HTML5
- Objective C (iPhone native development)
- Java (Android native development)
- jQuery Mobile

**Database Structure:**
Relational database
Django Framework
HTML Forms for Admin User Interface

**High Level Architecture Overview:**
See Appendix B for a physical diagram.

8. Development Plan and Project Management

8.1 Major Milestone Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Task</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>9/22</td>
<td>Project Initiated</td>
<td>Completed</td>
</tr>
<tr>
<td>10/3</td>
<td>System Requirements Review Presentation</td>
<td>Completed</td>
</tr>
<tr>
<td>10/7</td>
<td>Software development plan initiated, architecture plan and options to host database</td>
<td>Completed</td>
</tr>
<tr>
<td>10/18</td>
<td>Preliminary Design Review</td>
<td>Completed</td>
</tr>
<tr>
<td>10/24</td>
<td>Final design layout of application</td>
<td>Completed</td>
</tr>
<tr>
<td>11/14</td>
<td>Functioning sample application</td>
<td>Completed</td>
</tr>
<tr>
<td>11/15</td>
<td>Critical design review</td>
<td>Completed</td>
</tr>
<tr>
<td>11/16</td>
<td>Application and database development</td>
<td>Completed</td>
</tr>
<tr>
<td>1/30</td>
<td>Functioning application, pinging notifications, design and revision finalized</td>
<td>Completed</td>
</tr>
<tr>
<td>1/30</td>
<td>Functioning database, fully hosted and construction finalized</td>
<td>Completed</td>
</tr>
<tr>
<td>2/10</td>
<td>Properly functioning integration of system</td>
<td>Completed</td>
</tr>
<tr>
<td>2/11</td>
<td>Testing of Product</td>
<td>Completed</td>
</tr>
<tr>
<td>2/15</td>
<td>Product completed</td>
<td>Completed</td>
</tr>
<tr>
<td>2/20</td>
<td>Training package finalized</td>
<td>Completed</td>
</tr>
<tr>
<td>2/20</td>
<td>Train RMS employees</td>
<td>Completed</td>
</tr>
<tr>
<td>2/20</td>
<td>Deploy product</td>
<td>Completed</td>
</tr>
<tr>
<td>3/1</td>
<td>Collect data</td>
<td>Completed</td>
</tr>
<tr>
<td>4/1</td>
<td>Data Collection Completed</td>
<td>Completed</td>
</tr>
<tr>
<td>4/12</td>
<td>Data finalized, organized, analyzed and interpreted, recommended improvements.</td>
<td>Completed</td>
</tr>
<tr>
<td>4/18</td>
<td>Executive summary for RMS finalized</td>
<td>Completed</td>
</tr>
<tr>
<td>5/1</td>
<td>Design Day Presentation Completed</td>
<td>Completed</td>
</tr>
<tr>
<td>5/3</td>
<td>Raytheon Management Presentation</td>
<td>Completed</td>
</tr>
</tbody>
</table>

8.2 Project Management Update
See Appendix D for the Gantt Chart.

9. Budget and Suppliers

9.1 Budget Plan

| Description                                                                 | Quantity | Cost per Unit | Total Cost for Semester |
|-----------------------------------------------------------------------------|----------|---------------|-------------------------|-------------------------|

Page 32 of 82
### 9.2 Budget Analysis
The majority of the budget will be allocated to the database, Linode, and to the smart phones, which will be distributed to the participants in the Raytheon survey. Linode, will be used for 12 months, and costs $21.65 per month, for a 20GB of storage space. The smart phones will be purchased through Cricket Wireless, who will also be providing the service of the smart phones. Seven new ZTE Score phones will be purchased through Cricket’s website for $150.00 each. Each phone will have an unlimited data plan for $55.00 per month, with activation for two months. Other purchases include Wix, website builder, for $149.00, Logo Design Software for $49.95, and the poster board for the presentation, for $121.02. This all will total to $2258.78, leaving us under our budget and having extra money for any unexpected costs.

### 10. Acceptance Test Plan

The purpose of the Acceptance Test Plan is to verify and validate the high level functional requirements derived from our customer. In the verification phase of the Acceptance Test Plan, each functional requirement will be ensured that it would meet its criteria.

<table>
<thead>
<tr>
<th>Functional Requirement</th>
<th>Target Value</th>
<th>Method of Verification</th>
<th>Requirement Meet</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system shall have a device application where users will input their data</td>
<td>Meets Requirement Criteria</td>
<td>View database for input</td>
<td>Completed</td>
</tr>
<tr>
<td>The system shall have a simplistic user interface design</td>
<td>Meets Requirements Criteria</td>
<td>Testing of the application with actual users and receiving feedback</td>
<td>Completed</td>
</tr>
<tr>
<td>Administrator will have the ability to send a notification with a new question to all participants</td>
<td>Meets Requirement Criteria</td>
<td>Look for response on Client side from Server</td>
<td>Completed</td>
</tr>
<tr>
<td>The system shall have the application alert the user with notifications</td>
<td>Where 3g service is available to user</td>
<td>Time stamping of data</td>
<td>Completed</td>
</tr>
<tr>
<td>The system shall only recognize the inputs of the user by their User ID</td>
<td>Meets Requirement Criteria</td>
<td>Password verification of previously entered users into the database</td>
<td>Completed</td>
</tr>
<tr>
<td>The system will recognize a user not responding to an alert</td>
<td>Meets Requirement Criteria</td>
<td>Time Stamping of data sent to user and Time Stamping again given that the user does not respond a certain amount of time</td>
<td>Completed</td>
</tr>
</tbody>
</table>

**Requirement:** System shall have a device application where users will input their data.
To verify that this functional requirement was complete phones were distributed to users for testing. To test that the device was indeed transmitting the user inputted responses, a survey was sent out in which the users would respond to. If a response was recorded in the database it was determined that application met the requirement.
**Requirement:** The system shall have a simplistic user interface design.

To conclude that the system integrated a simplistic user interface there was a two phase testing period. The first test period was with the server side ensuring that was django was simple enough so that anyone could use it. The team decided that it was a very simple program and it should be used for the final design. The second test was on the client side determining if it was intuitive for the user on how to take the survey or input settings into the calendar page. From this test it was determined that a training packaged needed to be developed to eliminate any confusion that could occur while using the application. Therefore two training packages were created one for the android platform and the other for iPhone.

![Simplistic User Interface](image)

**Requirement:** The system shall have the application alert the user with notifications.

This requirement was completed during the testing phase of the application. During the test phase each of the group members were given a test phone to ensure that notifications were being sent so surveys could be taken. After several test runs and new versions of the application were developed the phone was successfully alerting the user via notifications.

![Notification](image)

**Requirement:** The system shall only recognize the inputs of the user by their User ID.

To ensure that no outside user could access the system a login page was developed on the application. Using Django, User IDs and Passwords could be set so that in order to gain access to the application a specific User ID and Password needed to be entered.

![Login](image)

**Requirement:** The system will recognize a user not responding to an alert.
Again during the test phase of the application responses were purposefully not responded to by the test subjects to view if the system would recognize a user not responding to a notification. It was determined after the testing phase that this requirement was completed via the time stamping of all notifications within Django. If a response was not given and another notification was sent to the user it could be viewed in the data that there was no response to a sent notification.

11. Conclusion

11.1 Data Analysis Conclusions
The QuiPST application was deployed within Raytheon and data was collected for two weeks, from April 9 to April 19. In this time, seven critical chain employees were involved in the data collection, using phones for the total or partial time of the study. With an average of 6 CC employees with 10 notifications each day for 9 days, there should have been 540 responses total, however only 294 were received.

After this data was collected and analyzed, it was found that a Raytheon employee working on a CC task spends more of their time working on this task than on a “non value added” task. This finding shows that the CC program is effective and shows a higher amount of “value added” or CC task time being higher than the previous study collected in the 1990s.

11.2 Application Conclusions
The QuiPST, Portable Survey Application, has been designed to meet the requirements specified by Raytheon. This client-server system will utilize the technologies of JQuery Mobile and PhoneGap to create an application that will work both on Android and iPhones as well as Django and Linode being used to create a relational database framework.

Specifically, Raytheon will have a device that they can implement within their high security specifications that will allow them to take surveys via a smart phone. The key feature that was
implemented was the random notifications that ping the user to take a survey. This feature met the requirement of collecting data in the most unobtrusive way, and did not alter the users responses.

Four main challenges arose when working on the QuiPST application. First, the system architecture had to be designed from the ground up, with no previous models or technical specifications. This allowed for innovative ideas with the use of Django web framework but also added additional testing and planning. Second, the client-server communication had to be determined before anything else could go further. This was ultimately accomplished through HTTP POST and GET functions. Third, the random pining implementation was a challenge in that a unique algorithm that generates a specific number of random notifications throughout the day, based on a percentage of likelihood depending on the time of day. This algorithm had to be tested and modified to get the exact randomness desired. Finally, the time constraint of this project, with the deployment and data collection included, left a 7 month span to plan, develop, test, deploy and analyze.

11.3 Project Conclusions
Overall, the importance of testing was shown throughout this project. Not only should testing be done as a preliminarily design step, but also for the exact product to be deployed to the company, in this case, Raytheon. Also, the importance of innovation and creative ideas was shown, first with the shift in scope of the project and then with the implementation of Django and finally in the use of Tasker and QB.

There are a few improvements that could be made to the system had there been more time and money provided. First, different Android cell phones would be purchased. Due to the current ones going to sleep at random times, several additions were made to the Android platform that was not necessary for the iPhone (QB and Tasker). One final improvement that could be made is implementing different survey formats using different features. Currently there is only a slider feature and radial buttons. Overall, project was under budget, but with upgrades on the cell phones, more money would be required.

Appendix A: Glossary

- **Administrator** - the person (or people) in charge of hosting the project. For the purposes of this project, Team 4943 will act as the administrator.
- **Application Programming Interface (API)** - source code used by software components used in communicating with each other. Includes several specifications such as variables, data structures, and object classes.
- **Asynchronous JavaScript and XML (AJAX)** - web development method used in creating asynchronous web applications. Web applications using AJAX are able to send and retrieve data from a server without interfering with the display and behavior of an existing page. The JavaScript may then update or modify the DOM of the HTML page.
- **Django** - a Python based open source web application framework. Used in this project to create and manage SQL databases.
- **Document Object Model (DOM)** - a language-independent, cross platform convention for representing and interacting with objects in HTML, XHTML, and XML documents.
- **GET (HTTP)** - request method supported by HTTP protocol. Requests a representation from the server (i.e. phone requests survey from server).
- **HyperText Markup Language (HTML)** - standardized markup language used to structure webpages.
- **HyperText Transfer Protocol (HTTP)** - network protocol used as the foundation for data communication on the world wide web.
- **jQuery Mobile** - HTML5 based user interface system that allows HTML code in PhoneGap look to the user like a native application.
- **JavaScript Object Notation (JSON)** - a subset of JavaScript
- **Notification** - also called a ‘push notification,’ a request initiated by the server to the smart phone prompting the user to begin taking a survey.
- **PhoneGap** - an HTML5 platform used to port HTML/JavaScript based code into different mobile frameworks.
- **POST (HTTP)** - request method supported by HTTP protocol. This method is called when the client needs to send data to the server as part of the request (i.e. submitting a completed survey form).
- **pull technology** - style of network communications wherein the initial data requests originate from the client and receives a response from the server.
- **survey** - a series of questions stored on the central server. The survey is stored as a tree of questions and subsequent questions depend on the previous answer.
- **Trigger** - when the server sends a request to a phone, it ‘triggers’ a notification to the user.
- **User** - anyone chosen to participate in a given survey. For this project, Raytheon employees on the Critical Chain will act as the user.
- **Virtual Private Server (VPS)** - a server that shares physical server hardware with other websites, but your website has a given amount of system resources
- **Python** - A programming language that allows for system integration efficiently.
- **Regular Expression (Regex)** - provides a concise and flexible means for "matching" (specifying and recognizing)strings of text, such as particular characters, words, or patterns of characters.

**Appendix B: Project Diagrams**

High Level Design Flow Chart 1.

High Level Design Flow Chart 2.
High Level Design Flow Chart 3.

System Overview Diagram.
Appendix C: Pseudo Code

Models Code
from django.db import models

class Group(models.Model):
    name = models.CharField(max_length=128)
    def __unicode__(self):
        return u"%s" % self.name

class UserID(models.Model):
    number = models.CharField(max_length=400)
    group = models.ForeignKey(Group)
    def __unicode__(self):
        return u"%s" % self.number

class Question(models.Model):
    question = models.CharField(max_length=400)
    group = models.ForeignKey(Group)
    def __unicode__(self):
        return u"%s" % self.question

class Answer(models.Model):
    answer = models.CharField(max_length=256)
    question = models.ForeignKey(Question, blank=True, null=True, related_name="next_question_set")
    def __unicode__(self):
        return u"%s %s to question %s" % (self.question, self.answer)

class Response(models.Model):
    user = models.ForeignKey(UserID)
    answer = models.ForeignKey(Answer)
    submitted = models.DateTimeField(auto_now=True)
    def __unicode__(self):
        return u"%s answered %s to question %s" % (self.user, self.answer, self.answer.question, question)

Views Code

from django.http import HttpResponse
from django.shortcuts import render_to_response, get_object_or_404

from models import Question, Group, Answer, UserID, Response

from django.http import HttpResponse

def test(request):
    return render_to_response('test.html')

def http_response(request):
    return HttpResponse('good')

def submit_answer(request):
    answer = get_object_or_404(Answer, pk=request.GET.get('answer'))
    user = get_object_or_404(UserID, number=request.GET.get('user'))
    Response.objects.create(
        user=user,
        answer=answer
    )
    if answer.next_question is not None:
        return render_to_response('survey.html', {'question': answer.next_question, 'user': user})
    else:
        return render_to_response('done.html')

def csv(request):
    responses = Response.objects.all()
    return render_to_response('csv.html', {'responses': responses}, mimetype='text/csv')

def survey(request):
    user = get_object_or_404(UserID, number=request.GET.get('user'))
    question = Question.objects.filter(next_question_set=None)[0]
    return render_to_response('survey.html', {'question': question, 'user': user})
Submit Answer

```python
def submit_answer(request):
    answer = get_object_or_404(Answer, pk=request.GET.get('answer'))
    user = get_object_or_404(UserID, number=request.GET.get('user'))

    Response.objects.create(  
        user=user,  
        answer=answer  
    )

    if answer.next_question is not None:
        return render_to_response('survey.html', {  
            'question': answer.next_question,  
            'user': user  
        })
    else:
        return render_to_response('done.html')
```

Calendar Code

```html
<!DOCTYPE html>
<html>
<head>
    <title>QnA Calendar</title>
    <!-- Code for jQuery mobile (JS) and jQuery (JS) -->
    <link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.css"/>
    <link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.css"/>
    <link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.css"/>
    <link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.css"/>
    <link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.css"/>
</head>

<body>

    <!-- Header -->
    <h1>Calendar</h1>

    <!-- Dropdown to select time for event -->
    <div data-role="dropdown" data-theme="b">  
        <div data-role="select" data-theme="b">  
            <label for="time">Select Time</label>
            <select id="time" name="time">
                <option value="08:00 AM">08:00 AM</option>
                <option value="09:00 AM">09:00 AM</option>
                <option value="10:00 AM">10:00 AM</option>
                <option value="11:00 AM">11:00 AM</option>
                <option value="12:00 PM">12:00 PM</option>
                <option value="01:00 PM">01:00 PM</option>
                <option value="02:00 PM">02:00 PM</option>
                <option value="03:00 PM">03:00 PM</option>
                <option value="04:00 PM">04:00 PM</option>
                <option value="05:00 PM">05:00 PM</option>
            </select>
        </div>
    </div>

    <!-- Dropdown to select day for event -->
    <div data-role="dropdown" data-theme="b">  
        <div data-role="select" data-theme="b">  
            <label for="day">Select Day</label>
            <select id="day" name="day">
                <option value="Monday">Monday</option>
                <option value="Tuesday">Tuesday</option>
                <option value="Wednesday">Wednesday</option>
                <option value="Thursday">Thursday</option>
                <option value="Friday">Friday</option>
                <option value="Saturday">Saturday</option>
                <option value="Sunday">Sunday</option>
            </select>
        </div>
    </div>

    <!-- Submit button -->
    <input type="button" value="Submit" />
</body>
</html>
```
Welcome Screen Code

```html
<!-- Welcome Screen Code -->

```

Pinging Algorithm Code

```bash
# Pinging Algorithm Code

```
### Appendix D: Project Schedule

#### System Requirements:

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Number of Data points Defined</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Design document requirements (Template)</td>
<td>Fri 10/7/11</td>
<td>Fri 10/7/11</td>
</tr>
<tr>
<td>Investigate options to host database</td>
<td>Tue 10/11/11</td>
<td>Tue 10/11/11</td>
</tr>
<tr>
<td>Software Development Plan, Deliverable to RMS</td>
<td>Mon 10/17/11</td>
<td>Mon 10/17/11</td>
</tr>
<tr>
<td>Informal Preliminary Design Review</td>
<td>Tue 10/18/11</td>
<td>Tue 10/18/11</td>
</tr>
</tbody>
</table>

#### Preliminary Design:

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Number of Data points Defined</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Design document requirements (Template)</td>
<td>Fri 10/7/11</td>
<td>Fri 10/7/11</td>
</tr>
<tr>
<td>Investigate options to host database</td>
<td>Tue 10/11/11</td>
<td>Tue 10/11/11</td>
</tr>
<tr>
<td>Software Development Plan, Deliverable to RMS</td>
<td>Mon 10/17/11</td>
<td>Mon 10/17/11</td>
</tr>
<tr>
<td>Informal Preliminary Design Review</td>
<td>Tue 10/18/11</td>
<td>Tue 10/18/11</td>
</tr>
</tbody>
</table>

#### Critical Design:

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Number of Data points Defined</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Design document requirements (Template)</td>
<td>Fri 10/7/11</td>
<td>Fri 10/7/11</td>
</tr>
<tr>
<td>Investigate options to host database</td>
<td>Tue 10/11/11</td>
<td>Tue 10/11/11</td>
</tr>
<tr>
<td>Software Development Plan, Deliverable to RMS</td>
<td>Mon 10/17/11</td>
<td>Mon 10/17/11</td>
</tr>
<tr>
<td>Informal Preliminary Design Review</td>
<td>Tue 10/18/11</td>
<td>Tue 10/18/11</td>
</tr>
</tbody>
</table>

#### Deployment:

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Number of Data points Defined</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Design document requirements (Template)</td>
<td>Fri 10/7/11</td>
<td>Fri 10/7/11</td>
</tr>
<tr>
<td>Investigate options to host database</td>
<td>Tue 10/11/11</td>
<td>Tue 10/11/11</td>
</tr>
<tr>
<td>Software Development Plan, Deliverable to RMS</td>
<td>Mon 10/17/11</td>
<td>Mon 10/17/11</td>
</tr>
<tr>
<td>Informal Preliminary Design Review</td>
<td>Tue 10/18/11</td>
<td>Tue 10/18/11</td>
</tr>
</tbody>
</table>

#### Data Analysis:

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Number of Data points Defined</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Design document requirements (Template)</td>
<td>Fri 10/7/11</td>
<td>Fri 10/7/11</td>
</tr>
<tr>
<td>Investigate options to host database</td>
<td>Tue 10/11/11</td>
<td>Tue 10/11/11</td>
</tr>
<tr>
<td>Software Development Plan, Deliverable to RMS</td>
<td>Mon 10/17/11</td>
<td>Mon 10/17/11</td>
</tr>
<tr>
<td>Informal Preliminary Design Review</td>
<td>Tue 10/18/11</td>
<td>Tue 10/18/11</td>
</tr>
</tbody>
</table>

#### Final Presentation:

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Number of Data points Defined</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Design document requirements (Template)</td>
<td>Fri 10/7/11</td>
<td>Fri 10/7/11</td>
</tr>
<tr>
<td>Investigate options to host database</td>
<td>Tue 10/11/11</td>
<td>Tue 10/11/11</td>
</tr>
<tr>
<td>Software Development Plan, Deliverable to RMS</td>
<td>Mon 10/17/11</td>
<td>Mon 10/17/11</td>
</tr>
<tr>
<td>Informal Preliminary Design Review</td>
<td>Tue 10/18/11</td>
<td>Tue 10/18/11</td>
</tr>
</tbody>
</table>

#### Appendix D: Project Schedule

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Number of Data points Defined</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Design document requirements (Template)</td>
<td>Fri 10/7/11</td>
<td>Fri 10/7/11</td>
</tr>
<tr>
<td>Investigate options to host database</td>
<td>Tue 10/11/11</td>
<td>Tue 10/11/11</td>
</tr>
<tr>
<td>Software Development Plan, Deliverable to RMS</td>
<td>Mon 10/17/11</td>
<td>Mon 10/17/11</td>
</tr>
<tr>
<td>Informal Preliminary Design Review</td>
<td>Tue 10/18/11</td>
<td>Tue 10/18/11</td>
</tr>
</tbody>
</table>

#### Critical Design:

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Number of Data points Defined</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Design document requirements (Template)</td>
<td>Fri 10/7/11</td>
<td>Fri 10/7/11</td>
</tr>
<tr>
<td>Investigate options to host database</td>
<td>Tue 10/11/11</td>
<td>Tue 10/11/11</td>
</tr>
<tr>
<td>Software Development Plan, Deliverable to RMS</td>
<td>Mon 10/17/11</td>
<td>Mon 10/17/11</td>
</tr>
<tr>
<td>Informal Preliminary Design Review</td>
<td>Tue 10/18/11</td>
<td>Tue 10/18/11</td>
</tr>
</tbody>
</table>

#### Deployment:

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Number of Data points Defined</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Design document requirements (Template)</td>
<td>Fri 10/7/11</td>
<td>Fri 10/7/11</td>
</tr>
<tr>
<td>Investigate options to host database</td>
<td>Tue 10/11/11</td>
<td>Tue 10/11/11</td>
</tr>
<tr>
<td>Software Development Plan, Deliverable to RMS</td>
<td>Mon 10/17/11</td>
<td>Mon 10/17/11</td>
</tr>
<tr>
<td>Informal Preliminary Design Review</td>
<td>Tue 10/18/11</td>
<td>Tue 10/18/11</td>
</tr>
</tbody>
</table>

#### Data Analysis:

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Number of Data points Defined</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Design document requirements (Template)</td>
<td>Fri 10/7/11</td>
<td>Fri 10/7/11</td>
</tr>
<tr>
<td>Investigate options to host database</td>
<td>Tue 10/11/11</td>
<td>Tue 10/11/11</td>
</tr>
<tr>
<td>Software Development Plan, Deliverable to RMS</td>
<td>Mon 10/17/11</td>
<td>Mon 10/17/11</td>
</tr>
<tr>
<td>Informal Preliminary Design Review</td>
<td>Tue 10/18/11</td>
<td>Tue 10/18/11</td>
</tr>
</tbody>
</table>

#### Final Presentation:

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Design</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Number of Data points Defined</td>
<td>Mon 10/3/11</td>
<td>Mon 10/3/11</td>
</tr>
<tr>
<td>Design document requirements (Template)</td>
<td>Fri 10/7/11</td>
<td>Fri 10/7/11</td>
</tr>
<tr>
<td>Investigate options to host database</td>
<td>Tue 10/11/11</td>
<td>Tue 10/11/11</td>
</tr>
<tr>
<td>Software Development Plan, Deliverable to RMS</td>
<td>Mon 10/17/11</td>
<td>Mon 10/17/11</td>
</tr>
<tr>
<td>Informal Preliminary Design Review</td>
<td>Tue 10/18/11</td>
<td>Tue 10/18/11</td>
</tr>
</tbody>
</table>
Critical Design:

<table>
<thead>
<tr>
<th>Critical Design</th>
<th>Days</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up dev.</td>
<td>3d</td>
<td>Wed 10/19/11</td>
<td>Fri 10/21/11</td>
</tr>
<tr>
<td>Design layout of app</td>
<td>3d</td>
<td>Mon 10/24/11</td>
<td>Wed 10/26/11</td>
</tr>
<tr>
<td>Write apps</td>
<td>3d</td>
<td>Thu 10/27/11</td>
<td>Mon 11/14/11</td>
</tr>
<tr>
<td>Critical Design review/Presentation</td>
<td>1d</td>
<td>Tue 11/15/11</td>
<td>Tue 11/15/11</td>
</tr>
<tr>
<td>&lt;CRITICAL DESIGN REVIEW&gt;</td>
<td>0d</td>
<td>Tue 11/15/11</td>
<td>Tue 11/15/11</td>
</tr>
</tbody>
</table>

Testing:

<table>
<thead>
<tr>
<th>Testing</th>
<th>Days</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test/revise apps</td>
<td>20d</td>
<td>Wed 10/12/11</td>
<td>Thu 3/1/12</td>
</tr>
<tr>
<td>Apps design &amp; functionality completed</td>
<td>23d</td>
<td>Fri 12/2/11</td>
<td>Thu 2/2/12</td>
</tr>
<tr>
<td>Database hosted</td>
<td>30d</td>
<td>Wed 10/12/11</td>
<td>Tue 11/22/11</td>
</tr>
<tr>
<td>Design of database finalized</td>
<td>12d</td>
<td>Tue 1/17/12</td>
<td>Wed 2/1/12</td>
</tr>
<tr>
<td>Construct database</td>
<td>10d</td>
<td>Mon 10/24/11</td>
<td>Fri 11/4/11</td>
</tr>
<tr>
<td>Test/revise database</td>
<td>18d</td>
<td>Tue 11/15/11</td>
<td>Wed 1/15/12</td>
</tr>
<tr>
<td>Database Completed</td>
<td>15d</td>
<td>Fri 12/2/11</td>
<td>Mon 2/21/12</td>
</tr>
<tr>
<td>Properly Tested/Revised For Integration of System</td>
<td>94d</td>
<td>Tue 12/6/11</td>
<td>Tue 2/23/12</td>
</tr>
<tr>
<td>&quot;Tree&quot; of Tasks questions completed &amp; delivered</td>
<td>27d</td>
<td>Mon 1/16/12</td>
<td>Tue 2/23/12</td>
</tr>
<tr>
<td>&lt;PRODUCT COMPLETED&gt;</td>
<td>0d</td>
<td>Thu 3/1/12</td>
<td>Thu 3/1/12</td>
</tr>
</tbody>
</table>

Deployment:

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Days</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draft of training package completed</td>
<td>6d</td>
<td>Thu 3/1/12</td>
<td>Thu 3/1/12</td>
</tr>
<tr>
<td>Training powerpoint</td>
<td>1d</td>
<td>Fri 3/2/12</td>
<td>Fri 3/2/12</td>
</tr>
<tr>
<td>Train RMS employees</td>
<td>1d</td>
<td>Mon 3/5/12</td>
<td>Mon 3/5/12</td>
</tr>
<tr>
<td>Demo, Info captured &amp; participant ID assigned</td>
<td>1d</td>
<td>Tue 4/10/12</td>
<td>Mon 4/9/12</td>
</tr>
<tr>
<td>Deployment within Raytheon</td>
<td>26d</td>
<td>Tue 3/4/12</td>
<td>Mon 4/9/12</td>
</tr>
<tr>
<td>&lt;DEPLOY PRODUCT&gt;</td>
<td>0d</td>
<td>Tue 4/10/12</td>
<td>Tue 4/10/12</td>
</tr>
</tbody>
</table>
Data Analysis:

<table>
<thead>
<tr>
<th>Task</th>
<th>Effort</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Analysis</td>
<td>60%</td>
<td>Wed 2/22/12</td>
<td>Mon 4/30/12</td>
</tr>
<tr>
<td>Collect data</td>
<td>70%</td>
<td>Tue 3/27/12</td>
<td>Tue 4/24/12</td>
</tr>
<tr>
<td>Data collected &amp; organized</td>
<td>20%</td>
<td>Wed 4/25/12</td>
<td>Wed 4/25/12</td>
</tr>
<tr>
<td>Data analyzed &amp; interpreted</td>
<td>5%</td>
<td>Thu 4/26/12</td>
<td>Thu 4/26/12</td>
</tr>
<tr>
<td>Recommended improvements</td>
<td>5%</td>
<td>Fri 4/27/12</td>
<td>Fri 4/27/12</td>
</tr>
<tr>
<td>Product Documents</td>
<td>5%</td>
<td>Mon 4/30/12</td>
<td>Mon 4/30/12</td>
</tr>
</tbody>
</table>

Final Presentation:

<table>
<thead>
<tr>
<th>Task</th>
<th>Effort</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of project completed</td>
<td>0%</td>
<td>Tue 5/1/12</td>
<td>Tue 5/1/12</td>
</tr>
<tr>
<td>High level timeline &amp; changes</td>
<td>0%</td>
<td>Tue 5/1/12</td>
<td>Tue 5/1/12</td>
</tr>
<tr>
<td>Ex. Summary comp.</td>
<td>0%</td>
<td>Tue 5/1/12</td>
<td>Tue 5/1/12</td>
</tr>
<tr>
<td>&lt;FINAL REPORT &amp; PACKAGE</td>
<td>0%</td>
<td>Tue 5/1/12</td>
<td>Tue 5/1/12</td>
</tr>
<tr>
<td>RMS presentation created</td>
<td>0%</td>
<td>Tue 5/1/12</td>
<td>Tue 5/1/12</td>
</tr>
<tr>
<td>RMS presentation rehearsal</td>
<td>0%</td>
<td>Mon 4/30/12</td>
<td>Mon 4/30/12</td>
</tr>
<tr>
<td>Presented to RMS</td>
<td>0%</td>
<td>Fri 4/27/12</td>
<td>Fri 4/27/12</td>
</tr>
<tr>
<td>Final touch up and rehearsal</td>
<td>0%</td>
<td>Mon 4/30/12</td>
<td>Tue 5/1/12</td>
</tr>
<tr>
<td>completed for class</td>
<td>0%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESIGN DAY PRESENTATION COMPLETED</td>
<td>0%</td>
<td>Tue 5/1/12</td>
<td>Tue 5/1/12</td>
</tr>
<tr>
<td>Written By</td>
<td>Personal Description of Individual Role</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Natasha Dennison</td>
<td>I had two major contributions to this report; they included the budget and the project management partitions. For the budget portion, I had to make sure our team was under budget, even after our extended services with Linode and Cricket, as well as other purchases we had throughout the year. In the beginning of this project we had decided to purchase a six-month plan with Linode, however we decided to extend it to a year, this way our sponsor could continue the use Linode as their server. Another big payment that had to be finalized were the cell phones, we originally paid for the phones until the end of April, however after talking with our sponsor we decided to pay for an extra month of service, until the end of May. The other partition of the report I was responsible for was the project management section. In the beginning of the year we started a “tentative” schedule for our project, however as the year went along our schedule started to change due to unexpected events. I fixed our project management worksheet to follow a more solid schedule, with the help of one of our sponsors, from there I was able to make a gantt chart and use these in our appendices.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Roxy Cruz</td>
<td>My specific duties involved assisting in development of the different technical aspects of the project. I developed the iPhone client application for QuiPST, and assisted in making the Python server-side scripts.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rae Gargione</td>
<td>As team leader, my main role in the report was to oversee all sections and delegate sections out to group members. Personally, I worked on Section 7- “How it works” and section 11- “Conclusion.” I also edited all sections of the report and formatted the entire report. In general, I was responsible for keeping up on the due dates of all aspects of the class, and work on any administrative tasks involved for our project. Also, I was responsible for organizing all information and working on the final touches of each part of the application and documentation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dimuth K.</td>
<td>I was responsible for the technical aspects of the project. I decided on the system architecture, and made the lead decisions on where to use push notification technology vs. server polling, where to use native development vs. making interactions browser based (globally applicable), and generally how to drive and govern the nature of the QuiPST system. I had a direct involvement in most code-based aspects of the project. I developed both Android phone applications for the QuiPST client package, and assisted my fellow developer in making the iPhone application. I helped developed the server side framework of QuiPST, and worked with the Python modules and server-based components that collected the required data and allowed for administrative actions. I came up with the pinging algorithm that notified users based on their work schedules and randomness. Finally I also wrote the HTML-based web pages that displayed the actual surveys and calendars to the end user, and wrote the java-script functions that made the calendar and surveys work properly.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Leo                  | The role in my team was primarily focused on the conceptual design of the project. I’ve created the requirements needed to understand the customer needs and make sure they
| Montoya | were implemented in the long run of the project. The secondary roles I had with the team is to be the contact person between our sponsor and the team, and design an informational website about our project’s design, features and functionality. Also, I was the lead on the website creation as well as the poster design this semester. |
| Luke Seavitt | Throughout the project I helped with a variety of different tasks. As a SIE major I helped develop the requirements document as well as the Con Ops report. As the project progressed and the technical level increased I focused on the development of the database with Rae. Once the database and the application were communicating I then took on the task of developing the training packages for each platform for the Raytheon users. During the collection phase I was then assigned with my main task for the project, data analysis. I generated the statistics as well as a confidence interval from the collected data (close to 300 responses) in order to give a conclusion to RMS if the programs they have implemented were actually working. Finally, I acted as one of the primary editors for reports, presentations, etc. |
## Appendix F: System Requirements

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>Description</th>
<th>Parent</th>
<th>Source</th>
<th>Status</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Functional</td>
<td>The system shall have a device application where users can input their data</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Suggested</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Functional</td>
<td>The system shall have a user setup</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Suggested</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Functional</td>
<td>The application will allow the users to setup up with different options</td>
<td>2</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Suggested</td>
</tr>
<tr>
<td>4</td>
<td>Functional</td>
<td>Users shall have the ability to alter the notification from a tone to a vibration</td>
<td>3</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Suggested</td>
</tr>
<tr>
<td>5</td>
<td>Functional</td>
<td>Users shall have the ability to select the hours and days for notification</td>
<td>3</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Suggested</td>
</tr>
<tr>
<td>6</td>
<td>Functional</td>
<td>Users shall have the ability to turn off alerts on any given day</td>
<td>3</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Suggested</td>
</tr>
<tr>
<td>7</td>
<td>Functional</td>
<td>The system shall allow the users to change the setup of application at any time</td>
<td>2</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Suggested</td>
</tr>
<tr>
<td>8</td>
<td>Functional</td>
<td>The system shall request demographic information from the user</td>
<td>Raytheon</td>
<td>Approved</td>
<td>Must</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Functional</td>
<td>Demographic information can be captured outside of the database</td>
<td>8</td>
<td>Raytheon</td>
<td>Approved</td>
<td>Must</td>
</tr>
<tr>
<td>10</td>
<td>Functional</td>
<td>The system shall ask survey questions</td>
<td>8</td>
<td>Team 4943</td>
<td>Approved</td>
<td>Must</td>
</tr>
<tr>
<td>11</td>
<td>Functional</td>
<td>The questions will be dynamic</td>
<td>10</td>
<td>Team 4943</td>
<td>Approved</td>
<td>Must</td>
</tr>
<tr>
<td>12</td>
<td>Functional</td>
<td>The questions can change or be modified at any time during the deployment of the survey life</td>
<td>10</td>
<td>Team 4943</td>
<td>Proposed</td>
<td>Must</td>
</tr>
<tr>
<td>13</td>
<td>Functional</td>
<td>The demographic survey should obtain an option for the user to correct or reenter data if needed</td>
<td>8</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Must</td>
</tr>
<tr>
<td>14</td>
<td>Functional</td>
<td>The system shall allow administrator to setup certain characteristics of the application</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Suggested</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Functional</td>
<td>The administrator can select a range for the number of random alerts given within the hours specified by each user</td>
<td>14</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Suggested</td>
</tr>
<tr>
<td>16</td>
<td>Functional</td>
<td>The administrator can select specific days the alerts will be sent</td>
<td>14</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Must</td>
</tr>
<tr>
<td>17</td>
<td>Functional</td>
<td>The administrator can select the minimum and maximum time between successive alerts</td>
<td>14</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Must</td>
</tr>
<tr>
<td>18</td>
<td>Functional</td>
<td>Administrator will have the ability to send out a trigger and a new question to all participants</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Must</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Functional</td>
<td>Administrator could initiate an instant poll by sending out an immediate alert and all participants will be alerted and prompted to answer the question(s)</td>
<td>18</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Desired</td>
</tr>
<tr>
<td>20</td>
<td>Functional</td>
<td>Administrator could set up a trigger to be sent out at a predetermined day and time, such as 9:00AM on this Friday</td>
<td>18</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Desired</td>
</tr>
<tr>
<td>21</td>
<td>Functional</td>
<td>The system shall have the application alert the user with notifications</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Suggested</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Functional</td>
<td>There will be two alert mode options</td>
<td>21</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Must</td>
</tr>
<tr>
<td>23</td>
<td>Functional</td>
<td>Notifications are triggered by the selected time intervals from the administrator, but overall are random</td>
<td>22</td>
<td>Raytheon</td>
<td>Proposed</td>
<td>Suggested</td>
</tr>
<tr>
<td>Page</td>
<td>Functional</td>
<td>Raytheon Proposed</td>
<td>Must</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>-------------------</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>User will decide when to open the application and answer questions</td>
<td>22</td>
<td>Raytheon Proposed</td>
<td>Must</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Each notification is composed of certain questions</td>
<td>21</td>
<td>Raytheon Proposed</td>
<td>Suggested</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>The system shall only recognize the inputs of the user by their User ID</td>
<td>Raytheon Proposed</td>
<td>Suggested</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Each user shall have a specific ID number</td>
<td>26</td>
<td>Raytheon Proposed</td>
<td>Suggested</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>The system will recognize a user not responding to an alert</td>
<td>Raytheon Proposed</td>
<td>Suggested</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>The system shall automatically remove the first alert after second alert has been established</td>
<td>28</td>
<td>Raytheon Proposed</td>
<td>Suggested</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>The system shall log the difference between a user initiated alert and a randomly answered alert</td>
<td>28</td>
<td>Raytheon Proposed</td>
<td>Must</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>The system shall log the time and date when first alert was initially sent</td>
<td>30</td>
<td>Raytheon Proposed</td>
<td>Suggested</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>The system shall log the time it takes the user to respond to survey</td>
<td>30</td>
<td>Raytheon Proposed</td>
<td>Suggested</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>The system shall log the previous alert as a void when the next alert has arrived and previous alert has not been recorded</td>
<td>30</td>
<td>Raytheon Proposed</td>
<td>Must</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Page</th>
<th>Technology</th>
<th>Raytheon Proposed</th>
<th>Must</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>The system shall consist of a smart phone</td>
<td>200</td>
<td>Raytheon Proposed</td>
</tr>
<tr>
<td>201</td>
<td>Smart phone platform will be Android and iPhone</td>
<td>200</td>
<td>Raytheon Proposed</td>
</tr>
<tr>
<td>202</td>
<td>Smart phone platform will be blackberry</td>
<td>200</td>
<td>Team 4943 Proposed</td>
</tr>
<tr>
<td>203</td>
<td>The system shall consist of a database</td>
<td>Raytheon Proposed</td>
<td>Suggested</td>
</tr>
<tr>
<td>204</td>
<td>There is a platform to host the server</td>
<td>203</td>
<td>Raytheon Proposed</td>
</tr>
<tr>
<td>205</td>
<td>Database will have the ability to have an internal and external factor</td>
<td>203</td>
<td>Raytheon Proposed</td>
</tr>
<tr>
<td>206</td>
<td>Internal will contain demographic information and be only accessible to those with specific clearance</td>
<td>205</td>
<td>Raytheon Proposed</td>
</tr>
<tr>
<td>207</td>
<td>External factor will contain data points collected from the use of the application and be accessible to our team</td>
<td>205</td>
<td>Raytheon Proposed</td>
</tr>
<tr>
<td>208</td>
<td>The database will be secured</td>
<td>202</td>
<td>Raytheon Proposed</td>
</tr>
<tr>
<td>209</td>
<td>The database will have a queue for multiple data submissions coming in at the same time</td>
<td>202</td>
<td>Raytheon Proposed</td>
</tr>
<tr>
<td>210</td>
<td>The database will log all answers—even if that answer is a “null” (the user did not answer the notification)</td>
<td>202</td>
<td>Raytheon Proposed</td>
</tr>
<tr>
<td>211</td>
<td>Database shall be capable of sending data to application</td>
<td>202</td>
<td>Raytheon Proposed</td>
</tr>
<tr>
<td>212</td>
<td>Application shall obtain questions from the user in a hierarchy tree format from the database</td>
<td>202</td>
<td>Raytheon Proposed</td>
</tr>
<tr>
<td>213</td>
<td>The hierarchy tree will be dynamic</td>
<td>211</td>
<td>Team 4943 Proposed</td>
</tr>
<tr>
<td>214</td>
<td>The hierarchy tree will have a template that the Administrator must fill in before they being the survey</td>
<td>211</td>
<td>Team 4944 Proposed</td>
</tr>
<tr>
<td>215</td>
<td>The system shall have an application</td>
<td>Raytheon Proposed</td>
<td>Suggested</td>
</tr>
<tr>
<td>216</td>
<td>The application is easily transferable from an Android phone to an iPhone</td>
<td>215</td>
<td>Raytheon Proposed</td>
</tr>
<tr>
<td>217</td>
<td>Technology</td>
<td>The application must be installable by each user via their own smart phone platforms</td>
<td>215</td>
</tr>
<tr>
<td>218</td>
<td>Technology</td>
<td>Android Market</td>
<td>217</td>
</tr>
<tr>
<td>219</td>
<td>Technology</td>
<td>iPhone App Store</td>
<td>217</td>
</tr>
<tr>
<td>220</td>
<td>Technology</td>
<td>The application use will be secured</td>
<td>215</td>
</tr>
<tr>
<td>221</td>
<td>Technology</td>
<td>Application shall obtain questions from the user in a hierarchy tree format from the database</td>
<td></td>
</tr>
<tr>
<td>222</td>
<td>Technology</td>
<td>Application can create survey given instructions</td>
<td></td>
</tr>
<tr>
<td>223</td>
<td>Technology</td>
<td>The application shall be capable of sending information to central database and vice versa</td>
<td>215</td>
</tr>
<tr>
<td>224</td>
<td>Technology</td>
<td>Application will allow for a user to be designated into a ‘group’ which allow the application be used multiple times at once</td>
<td></td>
</tr>
<tr>
<td>225</td>
<td>Technology</td>
<td>There will be a numeric code that goes with the questions to designate what survey it is</td>
<td>224</td>
</tr>
<tr>
<td>226</td>
<td>Technology</td>
<td>There will be an identifier of the set of questions itself AND the instance of the survey (time)</td>
<td>224</td>
</tr>
<tr>
<td>227</td>
<td>Technology</td>
<td>The application will have a built in calendar</td>
<td></td>
</tr>
<tr>
<td>228</td>
<td>Technology</td>
<td>The system shall allow administrator to setup certain characteristics of the application</td>
<td>227</td>
</tr>
<tr>
<td>229</td>
<td>Technology</td>
<td>The administrator can select a range for the number of random alerts given within the hours specified by each user</td>
<td>227</td>
</tr>
<tr>
<td>230</td>
<td>Technology</td>
<td>The administrator can select specific days the alerts will be sent</td>
<td>227</td>
</tr>
<tr>
<td>Technology</td>
<td>The administrator can select the minimum and maximum time between successive alerts</td>
<td>227</td>
<td>Team 4943</td>
</tr>
<tr>
<td>231</td>
<td>Technology</td>
<td>The administrator can select when to start and stop data collection for each individual</td>
<td>227</td>
</tr>
<tr>
<td>232</td>
<td>Technology</td>
<td>Appearance</td>
<td>215</td>
</tr>
<tr>
<td>233</td>
<td>Technology</td>
<td>Back button on every screen in the event of input mistakes in the same place</td>
<td>232</td>
</tr>
<tr>
<td>234</td>
<td>Technology</td>
<td>User friendly on all platforms</td>
<td>232</td>
</tr>
<tr>
<td>235</td>
<td>Technology</td>
<td>Application will have an initial log in and only accept specific user IDS</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>Performance</td>
<td>Application should respond quickly from the input of the user</td>
<td></td>
</tr>
<tr>
<td>301</td>
<td>Performance</td>
<td>Less than 1 second transition from screen to screen</td>
<td>300</td>
</tr>
<tr>
<td>302</td>
<td>Performance</td>
<td>The database and application must be able to handle a significant amount of data that could be input into the database until successful transfer</td>
<td></td>
</tr>
<tr>
<td>303</td>
<td>Performance</td>
<td>The response will take less than 20 seconds for the user to answer</td>
<td></td>
</tr>
<tr>
<td>304</td>
<td>Performance</td>
<td>The application should run well on a wide range of smart phone operating systems</td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>Utilization</td>
<td>The system requires purchases</td>
<td>Raytheon</td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
<td>-------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>401</td>
<td>Utilization</td>
<td>New mobile platforms purchased for participants with data packages</td>
<td>400</td>
</tr>
<tr>
<td>402</td>
<td>Utilization</td>
<td>The system shall be usable by its participants</td>
<td>Raytheon</td>
</tr>
<tr>
<td>403</td>
<td>Utilization</td>
<td>The system shall provide a training package to the user</td>
<td>403</td>
</tr>
<tr>
<td>404</td>
<td>Utilization</td>
<td>The system shall have a simplistic user interface design</td>
<td>403</td>
</tr>
<tr>
<td>500</td>
<td>Trade-Off</td>
<td>The system shall have application security</td>
<td>Raytheon</td>
</tr>
<tr>
<td>501</td>
<td>Trade-Off</td>
<td>The system shall have secure and restrictive application downloads</td>
<td>500</td>
</tr>
<tr>
<td>502</td>
<td>Trade-Off</td>
<td>Application security secures the application usage after download has taken place</td>
<td>500</td>
</tr>
<tr>
<td>503</td>
<td>Trade-Off</td>
<td>The system shall collect data</td>
<td>Raytheon</td>
</tr>
<tr>
<td>504</td>
<td>Trade-Off</td>
<td>The system shall use direct SQL statements to interact with the database</td>
<td>503</td>
</tr>
<tr>
<td>505</td>
<td>Trade-Off</td>
<td>The system shall contact Drupal based management system using XML-RPC protocol which Drupal converts to relevant SQL statements</td>
<td>503</td>
</tr>
<tr>
<td>506</td>
<td>Trade-Off</td>
<td>The application shall post information through email, which is parsed and written to a database</td>
<td>503</td>
</tr>
<tr>
<td>600</td>
<td>Test</td>
<td>The system shall have two phases of testing</td>
<td>Raytheon</td>
</tr>
<tr>
<td>601</td>
<td>Test</td>
<td>The system shall incorporate a software debugging phase</td>
<td>600</td>
</tr>
<tr>
<td>602</td>
<td>Test</td>
<td>A system integration phase shall be implemented</td>
<td>600</td>
</tr>
<tr>
<td>603</td>
<td>Test</td>
<td>The system's database and application shall communicate</td>
<td>603</td>
</tr>
</tbody>
</table>
Appendix G: Training Packages

Android Training Package

Slide 1

QuiPST Training Package

Slide 2

QuiPST Introduction

Overview:
- QuiPST is a portable survey tool that is designed to be used by anyone as a generic survey application.
- For the purpose of RMS, it will be used as a way to investigate how their employees spend their time.

Document Purpose:
- There are two goals of this document:
  1. Explain to QuiPST users how to modify the user settings.
  2. Demonstrate how to take a survey once the user is notified.
Slide 3

Initial Login

Slide 4

Step 1: Unlock Phone

1. Place Finger over unlock button
2. Slide button to right across entire screen

Slide 5

Step 2: Select QuiPST icon

1. Locate QuiPST icon
2. Place finger over QuiPST icon
3. Tap icon to open QuiPST Application
Step 3: Login

1. Select User ID bar
2. Enter Assigned User ID
3. Select Password ID bar
4. Enter Assigned Password
5. Select Submit button to Login

Note: Incorrect Login

If User ID or password is incorrect, the following Error message will appear.
Access will not be granted until valid User ID & Password have been entered.
Contact Elle Warner at elle.warner@raytheon.com if password reminder is needed.

Calendar Setup
Slide 9

Calendar Setup Purpose:

- One of the goals of QuiPST is to be able to adjust to the user’s work schedule. This is especially useful because RMS allows employees to work flexible work schedules.
- The default time settings of the application are 9:00 a.m. – 5:00 p.m. Monday-Friday. If your schedule differs from the default settings, the following pages demonstrate how to adjust the times the tool is active to mirror your work schedule.
- Once these times are adjusted the system will remember the new settings.
- Should you not be at work for any given reason, by selecting the button “Don’t Alert Today” this will stop alerts for that day. The alerts will begin again the following day.

Slide 10

Step 1: Open Calendar
1. Select Launch Calendar button
2. Calendar Edit Page will appear

Slide 11

Step 2: Select Day to Edit
1. Tap “Select a day” button
2. Tap day of week to Edit
Step 3: Delete Default Time

1. Tap "Delete Selected Time" to remove default 9-5 setting
   *Note if Work schedule is 9-5 skip this step

Step 4: Select Day to Edit

1. Tap the "Select Start Time" bar
2. Tap "+" or "-" button to set desired Start Time
3. Once Start Time is correct tap "Set Time" button
4. Repeat steps 1-3 for Setting End Time

Option: Apply to Entire Week

The option to apply a new time setting(s) to every day in a week, if you have a consistent work schedule, is possible by clicking "Apply to Entire Week" button.
Step 5: Add Times to Schedule

1. Once Start and End Times are inputted Tap "Add to Scheduled Times"
   Note: Multiple times can be inputted i.e. (8AM - 12PM) (1PM - 6PM) to compensate for lunch
2. To input multiple times Add 1st group of times then repeat Steps 7 to Add 2nd group
   Note: Times Cannot Overlap

Step 6: View/Delete Scheduled Times

1. To view Scheduled Times Tap "Scheduled Times Bar"
2. To Delete a Scheduled Time Tap desired time to Delete
3. Tap the "Delete Selected Time" Bar to delete Scheduled Time

Step 7: Saving Scheduled Times

1. Repeat steps 2-6 for each day of the week
2. To save inputted results Tap the "Save" bar
3. Once they have been saved, go back to QuiPST Pro, located on your 'Home' screen of your phone to finish the process.
Step 8: Syncing Calendar

1. Tap "Sync Calendar" Button
   *Note: This will allow you to receive notifications
2. A notification at the bottom will appear when it has been synced

Option: Don’t Alert Today

The option to not receive notifications for the day is available. By checking this box, notifications will be turned off for that day.

Answer a Survey
Slide 21

Step 1: Receiving Notification

1. Now that the calendar settings have been set according to your work schedule, simply wait for a notification.
2. When a notification occurs the phone will vibrate and make a noise.
   • Note: The screen of the phone may still be black when this noise occurs. By clicking the button on the top of the phone (used also to turn the phone on/off) you can light up the screen and the unlock the phone.

Slide 22

Step 2: Opening the Survey

1. To begin the survey first unlock the phone as demonstrated on page 6
2. Tap the top of the screen and slide your finger towards the bottom of the phone (This opens the notification center)
3. Tap the 1st QuiPST notification to begin the survey

Slide 23

Step 2: Taking Survey

1. Survey Appears on New Page
2. Read Question
3. Tap desired response
4. Verify response is correct
5. Tap “Submit” button to submit the survey
Step 3: Slider Bar

1. Read Question
2. Tap slider button
3. Drag till desired time is shown
4. Verify desired time is shown
5. Tap “Submit” button

*Note: Slider bar may read 9 after submit button is selected. This is OK, your original inputted time will still be saved.

Complete Screen

1. When the survey is successfully submitted QuIPST will open a “Done” page for user confirmation
2. This indicates you have finished the survey
3. The process is then complete and you can go back to work.

Edit Phone Settings
Option: How to change Notification Tones

1. Unlock the phone
2. Tap “quick app” button (see below)
3. Tap “Settings” app
4. See next page

Note: Tones are already set & volume is at Full with Vibrate On

Slide 28

Option: How to change Notification Tones

5. Tap the bar labeled “Sound”
6. Tap the bar labeled “Notification ringtone”
   a. To put in silent mode tap “Silent mode” box
   b. To set volume of ringtone tap “Volume button”
7. Tap desired ringtone for notifications & press “ok”

Slide 29

Turning Phone On

1. To turn phone on press and hold the button located on the top of the phone

*NOTE: After long periods of idle time the phone may spontaneously turn itself off. Repeat step 1 if this occurs
Slide 30

**Silent Mode OFF**

When the phone is on, there is an option for 'Silent Mode.' If the phone is in this mode, you will be unable to hear the notifications.

1. To check and make sure you are not in silent mode, click and hold the button on the top of the phone.

A box will appear that has different options:

1. **Silent Mode** - Make sure this says 'Sound is ON'.
2. **Airplane mode** - Make sure this is OFF.
3. **Power Off** - Use this to turn off the phone.

---

Slide 31

**Check Daily Notification Status**

---

Slide 32

**Daily Notification Status Purpose:**

- If you are not sure if your phone/QuiPST is functioning properly or if you are not sure if you have gotten any notifications in a certain day, you have the ability to check this yourself.

This feature will show you:

- **Your 'Quota'** or the number of notifications you should receive in a day.
- **Your 'Pings so far'** or the number of notifications you have received today.
- **Your 'Ping History'** or a list of the times when you did get notifications so far that day.
- **Your 'Status'** or what your phone is doing at the current moment.
- **Your 'Cooldown left'** or the amount of time left before you may receive another ping.
- **Your 'Minutes to work with'** or the total number of minutes you are available that day.

If you have any issues with your phone, please check this log.
Step 1: Open File Manager

1. Tap the house icon to pull up the menu.
2. Locate and select the ‘File Manager’ icon.

Step 2: Reading log.txt File

1. Select the ‘HTML Viewer’
2. Reading ‘Pings so far’ will tell you how many notifications you have received today.

Slide 35
Reset User ID
Step 1: Reset User ID

1. When you are finished with your data collection period, select the "Reset User ID" from the QuiPST Home page.

Congratulations.

You have now set up the Quick Portable Survey Tool.

You should start to receive alerts during your next scheduled work time (including today). At each alert, you will be prompted to start the survey. Please start the survey *each* time you receive an alert and respond to the questions. Thank you for your time and effort to help us gather data and improve our company.

If you have any questions, or the tool is not functioning as desired, please contact Elle Warner.

iPhone Training Package

Slide 1

QuiPST Training Package

The Quick Portable Survey Tool
QuiPST Introduction

Overview:
• QuiPST is a portable survey tool that is designed to be used by anyone as a generic survey application
• It can also be used as a way to investigate how their employees spend their time

Document Purpose:
• There are two goals of this document:
  1. Explain to QuiPST user’s how to modify the user settings
  2. Demonstrate how to take a survey once the user is notified

Slide 3

Initial Login

Slide 4

Step 1: Unlock iPhone

1. Place finger over unlock button
2. Slide button to right across entire screen
Slide 5

Step 2: Select QuiPST icon

1. Locate QuiPST icon
2. Place finger over QuiPST icon
3. Tap icon to open QuiPST Application

Slide 6

Step 3: Login

1. Select User ID bar
2. Enter Assigned User ID
3. Select Password ID bar
4. Enter Assigned Password
5. Select Submit button to Login
6. Notification Appears upon successful Login

Slide 7

Note: Incorrect Login

If User ID or password is incorrect, the following Error message will appear. Access will not be granted until valid User ID & Password have been entered.

Contact Elle Warner at elle.warner@raytheon.com if password reminder is needed.
Upon first login of QuiPST, a notification may appear, asking if alerts, sounds, and icon badges may be allowed. Click OK.

Calendar Setup Purpose:

- One of the goals of QuiPST is to be able to adjust to the user’s work schedule. This is especially useful because WRS allows employees to work flexible work schedules.
- The default time settings of the application are 9:00 a.m. – 5:00 p.m. Monday-Friday. If your schedule differs from the default settings, the following pages demonstrate how to adjust the times the tool is active to mirror your work schedule.
- Once these times are adjusted, the system will remember the new settings.
- Should you not be at work for any given reason, by selecting the button “Don’t Alert Today” this will stop alerts for that day. The alerts will begin again the following day.
Step 1: Open Calendar
1. Select Calendar button
2. Calendar Edit Page will appear

Step 2: Select Day to Edit
1. Tap "Select a day" button
2. Drop down menu appears
3. Select day of week to Edit
4. Once desired day selected tap "Done" button

Step 3: Delete Default Time
1. Tap "Delete Selected Time" to remove default 9-5 setting
   *Note if Work schedule is 9-5 skip this step
Slide 14

Step 4: Select Day to Edit

1. Tap the “Select Start Time” bar
2. Tap “+” or “-” button to set desired Start Time
3. Once Start Time is correct tap “Set Time” button
4. Repeat steps 1-3 for setting End Time

Slide 15

Option: Apply to Entire Week

The option to apply a new time setting(s) to every day in a week, if you have a consistent work schedule, is possible by clicking “Apply to Entire Week” button.

Slide 16

Step 5: Add Times to Schedule

1. Once Start and End Times are inputted Tap “Add to Scheduled Times”
   Note: Multiple times can be inputted i.e. (8-12) (1-6) to compensate for lunch
2. To input multiple times Add 1st group of times then repeat step 1 to Add 2nd group
   Note: Times Cannot Overlap
Step 6: View/Delete Scheduled Times

1. To view Scheduled Times Tap “Scheduled Times Bar”
2. To Delete a Scheduled Time Tap desired time to delete
3. Tap the “Done” button
4. Tap the “Delete Selected Time” Bar to delete Scheduled Time

Step 7: Saving Scheduled Times

1. Repeat steps 2-6 for each day of the week
2. To save inputted results Tap the “Save” bar
3. Once they have been saved Tap the “Go to Home” button

Option: Don’t Alert Today

The option to not receive notifications for the day is available. By checking this box, notifications will be turned off for that day.
Answer a Survey

Slide 21

Step 1: Opening the Survey

1. Now that the calendar settings have been set according to your work schedule, simply wait for a notification.
2. When a notification appears it happens in the form shown.
   • Note: This means it is time to take a survey.
3. To take the survey tap the QuiPST icon and slide it to the right.
4. The QuiPST application will then open with the survey to answer.

Slide 22

Step 2: Taking Survey

1. Survey appears on new page.
2. Read Question.
3. Tap desired response.
   • Note: Questions may require you to scroll down on screen.
4. Verify response is correct.
5. Tap “Submit” button to submit the survey.
1. When the survey is successfully submitted, QuiPST will open a "Done" page for user confirmation.
2. This indicates you have finished the survey and you can go back to work.

**Complete Screens**

1. Open "Settings" Application
2. Select "Sounds"
3. Under silent, select Vibrate to be on or off.
4. If off, select a Text Tone from the list provided.

**Edit Notification Settings**

**Option: How to set up Vibrate or Tone**
Step 1: Reset User ID

1. When you are finished with your data collection period, select the “Reset User ID” from the QuiPST home page.
2. Click “OK” to the Warning message that appears to confirm, or “Cancel” to go back.

Congratulations.
You have now set up the Quick Portable Survey Tool.
You should start to receive alerts during your next scheduled work time (including today). At each alert, you will be prompted to start the survey. Please start the survey *each* time you receive an alert and respond to the questions. Thank you for your time and effort to help us gather data and improve our company.

If you have any questions, or the tool is not functioning as desired, please contact Elle Warner.
QuiPST Administrator Responsibilities

The “How To” Set Up A QuiPST Survey

The administrator is responsible for entering the allowed user IDs and passwords, setting up the survey questions and answers, and then extracting the data.

Entering User IDs and passwords

To keep your survey secure, the Administrator will have to enter in all the User IDs and passwords that will be allowed to access the specific survey group.

This can be done by clicking the “add” associated with “User ids”

Now you can add the user number and password. The number area will only accept numbers, where the password area will accept both numbers and letters. Each User ID has to be associated with a group, which in this case, is Critical Chain. Since more than one user will be added, clicking “save and add another” will allow you to add multiple users without leaving the screen.
Once this is done, only users with the correct User ID and password will be able to access the survey from their mobile phones.

**Setting up the Survey**

When you want to create a new survey, the following steps should be taken.

First, you need to create a group. This can be accessed through quipst.com/admin/survey. This group will be specific for the type of survey you are creating.

Once you have clicked on “Add” to add a Group, you can name the group and click “Save”

You can see that your group “Critical Chain” has been saved, and that now you have two groups in your queue. Now you need to return to the quipst.com/admin/survey page, which can be done by clicking on “Survey” in the upper left corner under “Django administration”.
Now you will want to click on the “add” that corresponds with Question.

You can type in the first question you want your survey to ask, and then select the group you want the question to go in. This can be done by clicking the drop down menu. In this case, we want to add it to the group we just made called “Critical Chain.” Click save and then return to the quipst.com/admin/survey again.

Now that you have a question, you want to assign all the possible answers to this question. This can be done by clicking “add” that corresponds to Answers.
Type one possible answer to this question. In this case, the answer is “Working on the CC task that was assigned to me”. Then, use the drop down arrow to associate this answer to the appropriate question.

If you want to associate another question with the answer you just typed in, you will click the green arrow next to this New Question field. A new window will pop up, and allow you to write a new question, and associate that question back to the group again.

In this case, if a user answered that they are “Working on the CC task assigned to me” then a follow up question would be “How long have you been doing what you are doing now?” Once again, the group that should be selected is Critical Chain. Click save.

This pop up window will close and you will be able to see that the “Next Question” field has your new question in it. Click save. You can see that your information has been saved.

Now if you return to quipst.com/admin/survey, you can continue the process above of adding more answers and questions to your survey “tree.”

For example, another answer to the question “What are you doing right now?” could be “In a meeting (not related to my CC task).” This can be done by clicking the “add” button associated with Answer. You can see that now in your drop down menu for “Next Question” you have now a list of the previous questions you have made and can reuse or create a new one.
Click save and repeat this as many times as you want. The survey has no maximum number of questions or answers.

A visual representation of the survey created above is shown below.

![Survey representation](image)

Note: If you want to add an answer to a question that will allow for a slider bar of time values to be how the user will enter the time they have been working on a specific task, you will simply type in “SLIDER_1-50”, in all capitals, which will output a slider on the phone or web page.

**Extracting the data**

When your data collection time is over, you can extract it from the server by going to the URL [http://backend.quipst.com/survey/csv](http://backend.quipst.com/survey/csv). This link will immediately prompt you to ‘Save As” and click “Okay.”

![Extracting data](image)

Once this downloads, the file type must be changed to .csv, which means comma separated value. This can be done by simply right clicking on the file name when it appears under “Downloads” and selecting “Open Containing Folder”. Once this is open, the file will have and ending of .part, but this needs to be changed to .csv.
Note: A warning message may appear when trying to change the file type, but ignore this, click “Yes”, and proceed.

Now the file can be opened in excel and all the data will be available for analysis, starting with UserID, Question, Answer, Ping Date/Time, Response Date/Time.
Appendix H: Website

Home Page

Functionality Page
About QuiPST

QuiPST, Quick Portable Survey Tool, is an application survey tool created by a design team of engineering students from the University of Arizona as a Senior Capstone Project.

This project was completed March 1, 2012 with a small testing group of a company in the Tucson area.

Design Team

From left to right:

- Systems Engineer: Leonardo Montes
- Computer Engineer: Himanik Kalasinghe
- Industrial Engineer: Luke Reinert
- Management Engineer: Natasha Dussault
- Industrial Engineer: Rux Ganguli
- Computer Engineer: Brecon Coor

Questions or Comments?

Please email us with any questions you have about the QuiPST application, our design team or our website at:

quipsproject@gmail.com

Thank you.
Appendix I: Team Pictures

From presentation 1 to 5....