PINQUU PHASE 3 FINAL REPORT
Table of Contents

1. Introduction ........................................................................................................... 5
   1.1 Purpose ............................................................................................................. 5
      1.1.1 Human Computer Interaction (HCI) ......................................................... 5
      1.1.2 Information Organization and Visualization (IOV) ................................. 5
      1.1.3 Content Analysis ....................................................................................... 5
      1.1.4 External Applications ............................................................................... 6
      1.1.5 Mobile Workers ....................................................................................... 6
      1.1.6 Isolation (Iso) Layer .............................................................................. 6
   1.2 Background ...................................................................................................... 6
      1.2.1 HCI ........................................................................................................... 6
      1.2.2 Visualization/IOV .................................................................................... 7
      1.2.3 Content Analysis .................................................................................... 7
      1.2.4 External Apps ........................................................................................... 7
      1.2.5 Mobile Workers ....................................................................................... 7
      1.2.6 Isolation Layer ....................................................................................... 8
      1.2.7 Visualization/IOV .................................................................................... 8
   1.3 Group Members .............................................................................................. 8
      1.3.1 HCI ........................................................................................................... 8
      1.3.2 Visualization/IOV .................................................................................... 8
      1.3.3 Content Analysis .................................................................................... 8
      1.3.4 External Apps ........................................................................................... 8
      1.3.5 Mobile Workers ....................................................................................... 9
      1.3.6 Isolation Layer ....................................................................................... 9
   1.4 Overview - This is an executive summary of the project and approach ............... 9

2. Conceptual Design ............................................................................................ 10
   2.1 Problem Definition .......................................................................................... 10
      2.1.1 Base Line Scenario .................................................................................. 10
      2.1.2 Key System Requirements ....................................................................... 11
   2.2 Initial Solution Concepts ............................................................................... 11
      2.2.1 Table of selected technologies ................................................................ 12
      2.2.2 Visionary Scenario ................................................................................. 12
   2.3 Conceptual Design ........................................................................................ 13
      2.3.1 Selection criteria based on visionary scenario ....................................... 13
3. System Tutorial and Usage Scenario ................................................................. 20
   3.1 Summary of Integrated User Interactions .................................................... 20
      3.1.1 HCI ........................................................................................................ 20
      3.1.2 Visualization/IOV ................................................................................. 20
      3.1.3 Content Analysis .................................................................................... 20
      3.1.4 External Apps ......................................................................................... 22
      3.1.5 Mobile Workers ..................................................................................... 22
   3.2 Usage Scenario (Illustrated by screen images and pictures of the equipment) ... 22
      3.2.1 HCI ........................................................................................................ 22
      3.2.2 Visualization/IOV ................................................................................. 22
      3.2.3 Content Analysis .................................................................................... 25
      3.2.4 External Apps ......................................................................................... 27
      3.2.4.1 Instant Messaging .............................................................................. 27
      3.2.4.2 Document Sharing .......................................................................... 34
      3.2.5 Mobile Workers ..................................................................................... 36
      3.2.6 Isolation Layer ....................................................................................... 39
4. Detailed Design .................................................................................................. 40
   4.1 HCI ................................................................................................................. 40
      4.1.1 Functionality and Interactions ............................................................... 40
      4.1.2 Screens .................................................................................................... 40
      4.1.3 Experimental Evaluation ....................................................................... 43
   4.2 Visualization/IOV .......................................................................................... 49
      4.2.1 Functionality ........................................................................................... 49
4.2.2 Components ................................................................. 49
4.2.3 Experimental Measurements........................................... 51
4.2.4 Software Architecture .................................................... 53
4.2.5 Software Modules and Status .......................................... 54
4.2.6 Future Work ..................................................................... 59
4.3 Content Analysis ............................................................... 59
  4.3.1 Functionality .................................................................... 59
  4.3.2 Components ................................................................. 60
  4.3.3 Experimental Measurements ........................................... 60
  4.3.4 Software Architecture .................................................... 62
  4.3.5 Software Modules and Status .......................................... 63
4.4 External Applications .......................................................... 63
  4.4.1 Functionality ................................................................... 63
  4.4.1.2 Document Sharing ...................................................... 65
  4.4.2 Components .................................................................. 65
  4.4.2.1 Instant Messaging ....................................................... 65
  4.4.2.2 Document Sharing ...................................................... 66
  4.4.3 Experimental Measurements ........................................... 66
  4.4.3.1 Instant Messaging ....................................................... 66
  4.4.3.2 Document Sharing ...................................................... 66
  4.4.4 Software Architecture .................................................... 67
  4.4.4.1 Instant Messaging ....................................................... 68
  4.4.4.2 Document Sharing ...................................................... 68
  4.4.5 Software Modules and Status .......................................... 69
  4.4.5.1 Instant Messaging ....................................................... 69
  4.4.5.2 Document Sharing ...................................................... 70
  4.4.6 Design Alternative Evaluation Chart ................................ 71
  4.4.7 Future Work ..................................................................... 71
  4.4.7.1 Instant Messaging ....................................................... 71
  4.4.7.2 Document Sharing ...................................................... 71
4.5 Mobile Worker .................................................................... 71
  4.5.1 Functionality ................................................................. 71
  4.5.2 Components and pictures ............................................... 72
  4.5.3 Hardware Architecture .................................................. 77
<table>
<thead>
<tr>
<th>4.5.4</th>
<th>Software Architecture</th>
<th>77</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5.5</td>
<td>Hardware Modules and Status</td>
<td>80</td>
</tr>
<tr>
<td>4.5.6</td>
<td>Software Modules and Status</td>
<td>81</td>
</tr>
<tr>
<td>4.5.6.1</td>
<td>Communication to Isolation Layer</td>
<td>81</td>
</tr>
<tr>
<td>4.5.6.3</td>
<td>Status Stone Software</td>
<td>82</td>
</tr>
<tr>
<td>4.5.6.4</td>
<td>Mobile GUI Application</td>
<td>83</td>
</tr>
<tr>
<td>4.6</td>
<td>Isolation</td>
<td>83</td>
</tr>
<tr>
<td>4.6.1</td>
<td>Functionality</td>
<td>83</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Formal Specification</td>
<td>84</td>
</tr>
<tr>
<td>4.6.3</td>
<td>Experimental Measurements</td>
<td>85</td>
</tr>
<tr>
<td>4.6.4</td>
<td>Software Modules and Status</td>
<td>85</td>
</tr>
<tr>
<td>4.7</td>
<td>Conclusions</td>
<td>86</td>
</tr>
<tr>
<td>4.7.1</td>
<td>Requirement Feature Table</td>
<td>86</td>
</tr>
<tr>
<td>4.7.2</td>
<td>Summary of Key Design Issues</td>
<td>88</td>
</tr>
<tr>
<td>4.7.3</td>
<td>Lessons Learned</td>
<td>91</td>
</tr>
<tr>
<td>5. Project Management</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Implementation and Integration Phase Results</td>
<td>95</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Task Dependency Chart</td>
<td>95</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Summary of Work Log Hours for Phase 3</td>
<td>98</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Summary by Group of Individual Student Contribution</td>
<td>98</td>
</tr>
<tr>
<td>5.2</td>
<td>Summary of Entire Project</td>
<td>103</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Ranked Issue List</td>
<td>103</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Summary of Work Log Hours for All Three Phase For Entire Class</td>
<td>106</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Reflections on Work Log Hours and Distribution</td>
<td>106</td>
</tr>
<tr>
<td>5.3</td>
<td>Suggestions for Improving Class</td>
<td>108</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Purpose

The overarching goal of this project was to research and develop a proof of concept product that is designed to increase future workers’ productivity and provide solutions for tasks that current technologies do not easily solve such as finding a new project to work on. The class was broken up into six different groups with specific tasks (described below in 1.1.1 through 1.1.6) and integrated into a single project.

1.1.1 Human Computer Interaction (HCI)

During this final phase of the project, the HCI group continued to ensure the following: (1) that the final implementation of the technology encompasses goals from the visionary scenario and (2) Pinquu’s web portal undergoes user testing to maximize its web usability. For the web portal’s usability, we specifically tested for the tasks of finding an expert within the system and finding a project that may be relevant to the user’s interests. To make sure that the web portal would not only adequately facilitate users in these tasks, but would be an intuitive and efficient experience, we performed several usability studies: think-aloud tests, heuristic evaluation, and Cogtool analysis.

1.1.2 Information Organization and Visualization (IOV)

With HCI group designs of the system, the purpose of IOV group is to implement the design into fully functional webpage of Pinquu. While major role of IOV is to deal with the actual web implementation, there exists a minor role of IOV group which is to work as a bridge between HCI group and the Isolation Layer group. Communication with HCI and Isolation Layer group is critical. HCI group and IOV group are very alike as both groups deal with the design of the system. With the initial design, there exists some modification to the design due to some technical difficulties. For this matter, it is very important to communicate with HCI group to decide which features to include in the final design. While IOV group deals with the front-end design, Isolation Layer group is the one which IOV need to communicate to get all data. Bridge between Isolation Layer and IOV group allows data extraction from the database to the webpage which user would see.

1.1.3 Content Analysis

At the direction of the HCI group, Content Analysis group was tasked two separate goals. The first goal of the Content Analysis group is to provide a visual search tool that extracts various topics from a body (or “corpus”) of documents that include files such as presentation and written reports (both formal and informal). Any document that is entered into the database (via the Isolation Layer, see 1.1.6 below) is analyzed by Lingpipe, a natural language processing program, to create a number of groups of related words called topics. These topics are then arranged by the visualization tool Birdeye and exported as an Adobe Flash file for inclusion in webpages, etc.

The second goal of Content Analysis is to leverage the visualization tools provided by Birdeye to create a people web. In this web, each person is a node and edges connect friends to indicate a close relationship. Both goals were largely fulfilled at the end of Phase 3.
1.1.4 External Applications

The purpose of having External Applications is to provide additional pre-written software that will help in communication between Kiva users. These are not embedded in the Pinquu webpage due to the short span of time that we had to implement them. Synchronization with the database has been tested and completed such that updates in the Kiva database will be reflected accordingly in these external applications.

We provide two different external applications: Instant Messaging (IM) and Document Sharing. The motivation behind the selection of these two external applications is the law of propinquity discussed in Phase One. In order to reduce the virtual distance between workers, especially those in different countries and time-zones, it is important to be able to communicate instantly via readily available and easy-to-use software. The IM application allows for instant communication across countries while the document sharing application allows for synched file storage within groups and other additional features described in the rest of the report.

1.1.5 Mobile Workers

The purpose of the Mobile Worker group is to provide clients with an efficient work-environment with access to the same information and utilities both inside and outside of the office. The focus of the group is scaling the mobility component of a work environment without hampering the productivity aspect. We provide a mobile application for Android users to access data within Kiva, and have demonstrated the usage of the pico-projector to display data onto nearby surfaces for increased visibility. Additionally, the Mobile Workers group provides software for multiple users to interact over Johnny Chung Lee’s virtual whiteboard, and screen sharing capabilities over the internet to facilitate inter-office communication. Finally, we have implemented a physical form of the Status stone aimed to provide users with a simple way to set their status across multiple instant messaging clients. With one swift motion, users can set their status to indicate their availability without being interrupted from their task-at-hand. More specific features of each component will be discussed in detail in this report.

1.1.6 Isolation (Iso) Layer

The purpose of Isolation Layer is to appropriately handle every request that comes into the database. When a user wants to make modifications to data or wants to view certain information, the user triggers a certain action which is sent as a request to the database. The Isolation Layer is essentially an access layer that takes in the requests and sends out the appropriate responses through XML format.

“Isolation Layer is a proxy between the database and the front end, which would isolate the database and control query traffic. This isolating layer would take in XML commands from any user, perform the necessary database transactions, and respond in an appropriately formatted manner. Thus each team would not need database experts and the database had a layer of optimization protecting it.”

1.2 Background

1.2.1 HCI

From the previous phases, HCI determined the needs of the users and visionary scenarios of a technology that could be used to aid these needs. HCI also designed the look-and-feel of the Pinquu website and mock-ups of those screens. This phase focused on testing the usability of the designed interactions from the previous phases.
1.2.2 Visualization/IOV

With current Kiva webpage, there exists limitation of features that we can implement based on the design. This encourages our group to create an entirely new standalone design. With the new design, Pniquu contains multiple pages which are:
- Login Page
- Master Page
- Home Page
- Profile Page
- Project Page
- Search Result Page
- BirdEye Result Page

These pages are implemented with the Visual Studio with utilizing C#, ASP.NET on back-end and CSS on the front-end. All data transfers are done through XML which allowed the request information as well as data extraction from the database through Isolation Layer.

1.2.3 Content Analysis

The Content Analysis group is aimed at using existing natural language processing (NLP) tools to condense numerous long documents down into a small number of topics, each of which contains numerous sub- and sub-sub- topics. In addition to creating these topics, the Content Analysis team is tasked with presenting this data in a visually appealing manner as a topic web. This topic web is aimed at helping a worker find a project that he or she is interested in and has the necessary expertise. In phases 1 and 2, the Content Analysis team researched and began implementing these technologies. In the third phase, we complete our integration into Pniquu. In addition to the topic web, the content analysis team aims to create a people web in order to help individuals find a relevant expert to help him or her with a problem.

1.2.4 External Apps

Our test database currently consists of Kiva users since the Kiva database simulates an entire company consisting of different groups of workers working on different projects. This is similar to our target audience that consists of people working in a company that has offices overseas and requires more effective means of communication.

This last phase, the Implementation and Integration Phase, is where all the subsystems are completed and merged into a final system. The external applications group has always been working separately from all other subsystems, apart from the Isolation Layer. We have to go through the Isolation Layer to attain user data and information to update our servers and clients.

We have completed and tested all features that we discussed in Phase Two.

1.2.5 Mobile Workers

The Mobile Worker group has worked separately from the other groups, communicating with the Isolation Layer only for database access needs. Most of our implementations were manifested in hardware form, with minimal interaction between tools other groups were developing. During the Implementation and Integration Phase, we were able to complete most features discussed in Phase Two, and successfully integrate the subsystems into the final system.
1.2.6 Isolation Layer

In any robust software system, the standard protocol is to have a single database access layer. Creating a database access layer prevents access to the database from multiple areas in the system. Having multiple places that can access the database can be a huge problem when there is a bug in the code. It will be painstakingly time consuming to find what and where the problem is if issues arise. With a single access protocol, we immediately know where the problem is if it is regarding the database. This protocol simplifies structure.

1.2.7 Visualization/IOV

1.3 Group Members

1.3.1 HCI
- Vikram Chatterji
- Chung-Yi Chi
- Nicole Fernandez
- Shoshana Holtzblatt
- Ray Luong
- Marcus Perez
- David Randall

1.3.2 Visualization/IOV
- Ajay Ghadiyaram
- Jaejoon Lee
- Jihoon Kim
- Paul Caravelli
- Paul Cho
- Rachita Chandra
- Suong-Sun Hong
- Rantao Chen

1.3.3 Content Analysis
- Steven Luminais
- Jason Lei
- Rohith Salim

1.3.4 External Apps
- Po Shin Huang
- Ivan Lee
- Cassie Li
- Skylar Roebuck
- Yiling Tay
- Yumin Wong
1.3.5 Mobile Workers
- Rohit Banerjee
- Rantao Chen
- Christopher Jo
- Daniel Lin
- Veeren Mandalia
- Rika Nakahara
- Thomas Tzou

1.3.6 Isolation Layer
- Kendra Garwin
- Skanda Mohan
- David Wang
- Andrew Yi

1.4 Overview - This is an executive summary of the project and approach

The Implementation and Integration Phase is where our various subsystems highlighted in the Detailed Design Phase are tested, completed and merged into a final system. The final report thoroughly illustrates the entire process of implementing each subsystem, including details on the software and hardware components.

Specifically, Section 2 describes the baseline scenario and some key system requirements based on it. After which, it presents the technologies, selected by each subsystem, and the visionary scenario. The conceptual design of each subsystem follows shortly after. Section 3 provides a tutorial on using the system and illustrates a usage scenario. Section 4 goes into the details of each subsystem, including functionality, components, experimental measures, and hardware and/or software architecture and modules. Section 5 focuses more on project management, detailing the amount of time contributed to different tasks for different subsystems, as well as suggestions for improving the class.
2. Conceptual Design

2.1 Problem Definition

2.1.1 Base Line Scenario

2.1.1.1 Target Users

<table>
<thead>
<tr>
<th>Distributed Software Engineer - Junior</th>
<th>John, 25 years old, software developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed Software Engineer - Senior</td>
<td>Sumit, 35 year old, software developer</td>
</tr>
<tr>
<td>Distributed Manager</td>
<td>Phillip, 40 year old, manager of a distributed development team</td>
</tr>
</tbody>
</table>

Current Scenario

John is a fairly new software developer and is part of a distributed software development team. He is currently working on the user interface for a health kiosk. As usual, today he gets to work, boots up for the day’s work, and launches Outlook, Sharepoint, Skype, and an internal Instant Messaging tool for communication. Before 9am, he is ready and continues working on the user interface. He runs into an issue integrating a glucose bio-sensor and needs some help. The first thing comes to his mind is that someone in the company might have integrated a similar sensor, so he opens Sharepoint as well as internal databases and searches files and documentation to see if there is relevant information that can help him instantly. Unfortunately, there is no effective/efficient way to get the answer. Then John tries to search through online forums and search engines like Google. However, due to the difficulty of explaining his specific problem very well, he doesn’t get any helpful answers. Unable to solve the problem on his own, he opens the I.M. window and sends an instant message to Sumit, a friend who is also a software developer in the department, for help. Nevertheless, Sumit is on a vacation and won’t be available in 5 days. Now, John doesn’t know whom to approach within the department to solve his problem. Therefore, he decides to post the problem on an online discussion forum and wait until someone responds and hopefully gives a satisfactory answer. The problem of no efficient way to find helpful information/experts disrupts his workflow seriously.

While waiting to hear back John goes onto SharePoint and the company wiki to gain a sense of other projects he may like to work on. John can’t really find anything that is interesting. He tells his manager, Phillip who is out on the road. Phillip cannot login and gain look through the projects until he gets back. Phillip will not be back until next week. John needs to find something to work on so he picks a project that he can do but is not very interesting. When Phillip comes back he looks through the company website and realizes John could have helped out on a project that would have really interested him. John missed out on finding about the project and the company didn’t get the full benefit of his skills. As a result the company may end up losing him as an employee because he is bored at work.
2.1.2 Key System Requirements

The system must address the following requirements:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Find an Expert</td>
</tr>
<tr>
<td>R2</td>
<td>Expose users expertise</td>
</tr>
<tr>
<td>R3</td>
<td>Knowledge and visibility of availability</td>
</tr>
<tr>
<td>R4</td>
<td>Enable rapid communication</td>
</tr>
<tr>
<td>R5</td>
<td>Improve search and make it easier to find prior project work</td>
</tr>
<tr>
<td>R6</td>
<td>Enable real-time communication and collaboration</td>
</tr>
</tbody>
</table>

2.2 Initial Solution Concepts

The Human-Computer Interaction team helped create the focus for this project by doing baseline research into the remote work and overcoming the law of propinquity when it comes to collaborative work. The research included interviews with five people who do work remotely with employees, employers, coworkers and clients. It also included a literature review of 17 articles related to remote and collaborative work, and meetings with Carnegie Mellon faculty and researchers working on topics related to remote workers.

Key findings from interviews included the following: (1) Instant messaging is core component of remote work. It helps not only enable real-time minimally invasive communication, but via availability status, instant messaging programs can communicate productivity and motivate coworkers. (2) Currently remote workers who dial into a real world meeting often lack the political power of workers that are at the location. Often it is easy for their key points and arguments to be lost. (3) Available technology can actively shape the way work can be done, particularly for smaller firm. The availability of a collaborative tool can dramatically impact the quality and quantity of work that gets completed.

Key findings from meetings with Carnegie Mellon faculty and researchers included the following: (1) Typically there are few good, central places for groups to communicate about their work. (2) Management and synthesis will be effective if handled through the cloud. (3) Current tools make it difficult to assign and monitor ownership of complex projects.

The findings from interviews, literature reviews and faculty contacts were compiled using an affinity diagram process and grouped into the initial solution concept and corresponding functional requirements listed below:
- Finding an expert.
- Personal profile management
- Visibility of expert proximity, interpretability and availability
- Instant messaging
- Intelligent and context-aware expert finder based on current work
- Content/Desktop sharing
- Finding a new project.
- Interactive wall of projects visualization
- Company news-feed
- Hubs (via Auto aggregation + hash tagging)
- Browse and search ability in the hub
- Tracking active members of the hub
- Monitor my team
- Team Screen: monitor that shows status, history, interpretability, location, assigned project teams, current work tasks, dependencies.
- Mobile Team screen
- Status Stone - quickly set status device
- IM delays messages based on employee status
- Employee profiles with current projects

While the initial design solution varied in the following phases for various reasons (e.g. the intelligent and context-aware expert finder based on current work was pushed to future plan given the current technical feasibility), the research, synthesis and ideation processes indeed provide valuable insights for our following design phases.

### 2.2.1 Table of selected technologies

Table 2.2.1.1 summarizes the selected technologies in each group and illustrates the purpose of the technologies.

<table>
<thead>
<tr>
<th>Group</th>
<th>Technology</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCI/GUI</td>
<td>Adobe Photoshop</td>
<td>Wireframing</td>
</tr>
<tr>
<td></td>
<td>Adobe Illustrator</td>
<td>Wireframing</td>
</tr>
<tr>
<td></td>
<td>OmniGraffle</td>
<td>Wireframing</td>
</tr>
<tr>
<td></td>
<td>CogTool</td>
<td>Cognitive analysis for interface use</td>
</tr>
<tr>
<td>Information Organization</td>
<td>Dynamic HTML</td>
<td>Website infrastructure</td>
</tr>
<tr>
<td>and Visualization</td>
<td>AJAX</td>
<td>Interactive website component</td>
</tr>
<tr>
<td>Text Extraction</td>
<td>LingPipe</td>
<td>Content analysis</td>
</tr>
<tr>
<td></td>
<td>BirdEye</td>
<td>Information visualization</td>
</tr>
<tr>
<td>External Application</td>
<td>iFolder</td>
<td>Document sharing</td>
</tr>
<tr>
<td></td>
<td>Spark (Windows, Linux)</td>
<td>Instant messaging Client</td>
</tr>
<tr>
<td></td>
<td>Adium (Mac)</td>
<td>Instant messaging server</td>
</tr>
<tr>
<td>Mobile Worker</td>
<td>Status stone</td>
<td>Availability Indication</td>
</tr>
<tr>
<td></td>
<td>Virtual whiteboard</td>
<td>Project display</td>
</tr>
<tr>
<td></td>
<td>Pico Projector</td>
<td>Mobile projector</td>
</tr>
<tr>
<td>Isolation Layer</td>
<td>Database</td>
<td>Data storage</td>
</tr>
<tr>
<td></td>
<td>XML</td>
<td>Protocol format</td>
</tr>
<tr>
<td></td>
<td>Git</td>
<td>Version control</td>
</tr>
</tbody>
</table>

Table 2.2.1.1 Selected Technologies.

### 2.2.2 Visionary Scenario

As the Rapid prototyping team explored the idea of remote work over the course of the semester, they developed three core visionary scenarios involving finding an expert, finding a new project and managing a small, remote team. The Human-Computer Interaction team came up with several technology ideas based on these scenarios, but the semester wore on and technical constraints became more pressing, certain technologies were discarded and the scenarios were narrowed to a single workflow: finding an expert.

In the scenario Kevin is a new employee working on glucose monitors and trying to
understand how the thresholds work so he can program part of the interface. He’s not sure conceptually what’s going on, and no one on his team seems to know, so he logs into Pinquu, the internal connection tool to see if he can find an expert. He first looks at the wall of topics to see what glucose sensor is related to. He can see it comes closes to bio sensor and sensor, so he tries these as search terms. In the search results he sees that Janice is an expert at both coding and bio sensors. He can see right in the interface that she is available. She has been maintaining her status using an external device, the status stone so he is confident the status is correct.

Kevin and Janice chat via IM and while she says she is qualified to help she doesn’t have enough time for a meeting. She recommends the bio sensors intra-company group as a source for other experts. Kevin checks this and sees that Ming is one of the top contributors. He contacts Ming, who explains what Kevin needs to know using a virtual white board and Kevin posts a file to the group’s page about it.

Later, Janice sees Kevin has posted a new document based on his conversation with Ming. This pulls up as part of her news feed. She is able to quickly scan the document and see that Kevin got the info he needed. Later, in a meeting Janice is asked about biosensors. She pulls up the document using Pinquu mobile and forwards it to the person who asked the question. She is also able to project the document using a pico projector.

2.3 Conceptual Design

2.3.1 Selection criteria based on visionary scenario

Based on the visionary scenarios, we determined several needs of the user that needed to be addressed:
• Finding an expert
• Finding a new project
• Create more transparency between the different teams in terms of what they have worked on in the past and what they are currently working on.

2.3.2 Product design specifications

Based on the visionary scenario, and exploratory work of the technical teams in finding and learning potential technologies, several of the explored technologies were selected for use in the future Pinquu system. These technologies include a Pinquu web portal, a mobile portal, the status stone device, a virtual whiteboard, pico micro projectors, a file sharing service and an instant messaging service. Selection criteria included price (free or minimally priced preferred), malleability and supportability (open-source and up-to-date preferred) as well as functionality.

2.3.3 User Interaction

During this final phase of the project, the HCI group continued to ensure the following: (1) that the final implementation of the technology encompasses goals from the visionary scenario and (2) Pinquu’s web portal undergoes user testing to maximize its web usability. For the web portal’s usability, we specifically tested for the tasks of finding an expert within the system and finding a project that may be relevant to the user’s interests. To make sure that the web portal would not only adequately facilitate users in these tasks, but would be an intuitive and efficient experience, we performed several usability studies: think-aloud tests, heuristic evaluation, and Cogtool analysis.
In usability studies, think-aloud tests assign users a task to accomplish using a system. They are asked to "think aloud" and express their thoughts as they use the system as though they were by themselves. The researcher simply writes notes and prompts the user to continue talking. By conducting a series of think-aloud tests, we were able to discover what novice users had trouble with while finding experts or finding an interesting project.

Finally, our heuristic evaluations of Pinquu evaluate the system based on 10 heuristics, or industry standards created by Jakob Nielsen. We evaluated the system’s usability along these heuristics and along the tasks that we designed for: finding an expert and finding a relevant project. The heuristic evaluation was a quick way for HCI experts to discover any glaring usability issues in the design.

We used Cogtool, a "cognitive crash dummy" software tool, to estimate efficiency of Pinquu for experts. Developed by Bonnie John at Carnegie Mellon, Cogtool aims to simulate human performance. This eliminates the need to find actual users, which can be a time-consuming and expensive task. Based on models of cognitive and motor processes, Cogtool can be used to estimate the time it takes an expert to complete the tasks we designed for in Pinquu.

2.3.4 Visualization

Based on the HCI design, Pinquu contains 7 main pages which are – Login, Master, Home, Profile, Project, Search Result, and BirdEye Result Page. While master page is the part of all other pages, each page is unique in a way that it contains specific information about the user himself or the projects. On the other hand, all pages are connected to each other by link to either the name of the project or the name of the user. The final design of Pinquu would be integrated to the current Kiva as a hyperlink which then can be used by all Kiva users. Here is the main structure of the Pinquu with its main features:
- Login Page
- Authentication of user
- Master Page
- Exists on every pages
- Search bar (Top)
- Login status (Top right corner)
- List of current projects (Left Panel)
- Calendar (Right Panel)
- Meetings (Right Panel)
- Home Page
- News feed
- Profile Page
- Status (Green = Available, Red = Unavailable)
- Contact Information
- Skills
- Documents
- Project Page
- Documents
- Messages
- Search Result Page
- Search results: People, Projects, Documents
- Link to Birdeye result
- BirdEye Result Page
- Graph chart of the search result
  All but Birdeye result for search result page would come from the Isolation Layer which is connected to the database. BirdEye result is in flash format which is created by content analysis group.

2.3.5 Content Analysis
After discussions with the HCI team, the Content Analysis is tasked with creating visual search tools incorporating people and topics generated from relevant documents. People and topics were arranged in a movable, circular arrangement where the user can move between topics/people with ease as well as view varying levels of detail.

2.3.6 External Applications

2.3.6.1 Instant Messaging
Our final choices of Instant Messaging (IM) clients, are Spark Messenger 2.5.8 for Windows/Linux and Adium 1.4.1 for Mac OSX. Pidgin Messenger, our initial choice, failed to add groups effectively even after an update to the server. It was likely that Pidgin had its own cache, which we had no access to. As such, we had to look for alternative IM clients that suit the user visionary scenarios in the Conceptualization Phase.
Adium was chosen for the Mac OSX platform as its user interface is greatly customized for Mac users. In addition, there was no problem adding Jabber accounts when we tried that in our user testing. Jabber uses the Extensible Messaging and Presence Protocol (XMPP), which ties in with our Openfire Server.
However, Adium has only been developed for the Mac OSX platform. As such, the natural secondary choice of an IM client for the Windows/Linux platform was Spark Messenger. Spark Messenger works the easiest with the Openfire server. But because of its lack of user interface features, we chose Adium over Spark for the Mac OSX platform. Both IM clients are able to perform the minimum requirements set in the visionary scenarios.
The IM client will be a separate application from the Pinquu webpage itself. In other words, it will not be embedded into the Pinquu webpage. It will be an external application that the user will have on his/her desktop to launch. This implementation was discussed in the Conceptualization Phase. We had determined that embedding the chat client into the webpage would require a higher level of coding expertise and a lot more time to implement, as well as a much more restricted feature set.
We have installed Openfire 3.6.4 as our server on the Linux OpenSUSE 11.2 desktop in the lab. Openfire will allow us to manage user accounts, create chat sessions, edit system properties and stay updated with the server information. In addition, Openfire links with the database that we specify, which is currently linked to the Kiva database via the Isolation Layer.

2.3.6.2 Document Sharing
In the Conceptualization Phase, the communication patterns group identified the sharing of documents to be of vital importance within the scope of the visionary scenarios. With the different document sharing functionality, the functionality deemed most important were the ease of synchronization, management of access rights, and the ability for group sharing. iFolder was identified as a clear winner as it was an open source solution that met the group’s requirements. iFolder is meant to handle the secure storage of individual user documents. The usernames are to
be synchronized with Kiva using software integration scripts. Additionally, integration scripts are used to maintain a list of iFolder file-path names and their associated user.

2.3.7 Mobile Worker

In the Conceptualization Phase the Mobile Worker group researched many hardware capabilities that are available in the market and decided which ones would be most useful for a worker of the future. This includes applications such as mobile video chat, portable projection systems, and ways to interact with others easily when away from the office. Technologies that were chosen include a portable virtual whiteboard as well as a compact projection system. Furthermore, the Mobile Worker group set out to build upon existing technologies to improve the office environment. For example, building a physical manifestation of an employee’s status for ease of use or creating a mobile application to access the Kiva were some ideas that we tackled as a group.

2.3.8 Software Architecture

Figure 2.3.8.1 Overall project software architecture highlighting the Isolation Layer in the center of many different features.

Figure 2.3.8.1 shows the Isolation layer in the center of the system. The isolation takes care of all the requests that come through all front end services and responses with appropriate information from the database.
Figure 2.3.8.2 The various functionality of the Dispatcher.

Figure 2.3.8.2 illustrates the functionality of the Dispatcher in the Isolation Layer. The Dispatcher contains distinct level of protocols between the front end and the back end in order to make the process secure and dependable.
2.3.9 Hardware Architecture

As shown in Figure 2.3.9.1, there is one primary desktop machine running OpenSUSE 11.2 which houses the server and code for Pinquu. There are also laptops which can represent on-site workers or mobile workers. Both machines will have a webcam with built-in microphones for teleconferencing. The Wiimote and infrared (IR) pens are used for the Virtual Whiteboard and the Status Stone is a physical device that can be flipped to change user status. The diagram for setting up the Virtual Whiteboard is shown below:

Figure 2.3.9.1 Hardware Architecture

All of these will be connected to a central router so testing can be done internally. The Pico Projector is able to be connected to a phone (iPhone) via a special cable or a laptop via a
VGA cable. This provides an alternative display for different devices and is especially useful in a mobile environment.
3. System Tutorial and Usage Scenario

3.1 Summary of Integrated User Interactions

3.1.1 HCI

User Interactions were a joint design between HCI and Visualization/IOV; please see the below section 3.1.2 for HCI’s User Interactions.

3.1.2 Visualization/IOV

A large number of the variations we made while implementing the mockup designs provided to us by HCI were due to realizations of user scenario shortcomings. We wanted to be sure that the GUI we were developing was intuitive yet had high functionality. In order to meet these objectives, we iteratively ran through the GUI from a user’s perspective to ensure there were no gaps in the user’s needs. Pinquu would provide much of information that one need to know about the company and its project through different pages such as Project Page, Profile Page and Search Result Page.

3.1.3 Content Analysis

3.1.3.1 People Web

The people web was designed to so that users could more easily see people related to them by expertise by displaying in a visually appealing way (Figure 3.1.3.1.1a). The look and feel was so that people could easily see links between people and their relationships. From the figure, we can see that nodes are not color coded, and there are edges linked to people if their relationship is amongst the top five calculated for that person.

Figure 3.1.3.1.1 a: Sample People web from Birdeye  b: Control bars on the side.

The main reasons for this simplistic model rather than a more complex model were ease of traversal and technical limitations. The way a user navigates is simply by double clicking a node. When a user does so, the graph shifts to have the selected node in the center allowing the user to see the relationships of that node. Additionally users can also click and drag nodes to rearrange nodes and inspect relationships. The graph also has controls on the side for different viewing options (3.1.3.1.1b). The Degrees of separation toggles the number of nodes you show beyond the start node, the Birdeye Zoom allows the user to zoom in and out of the graph, the
scaling bar affects the size of the nodes and the Refresh graph button resets the graph back to the initial state.

One limitation on not having nodes color coded, and having different sizes based on relationship was that we couldn’t dynamically set the size or color of the node based on which node was in the middle. Meaning a specific node would have to change size and color to display the specific relationship with the person in the center. Unless there was one generated graph for every individual person we would have to store every person’s interaction list within the graph XML file, so that depending on which node is the center, the nodes would change size accordingly. However, storing all that information within the graph made the solution not scalable and thus we stuck with the simplistic model.

3.1.3.2 Topic Web

Unlike the people web, the topic web seemed to have one definite start, which would be “Pinquu”, since it viewed all documents on the website to generate a topic model of the entire project (Figure 3.1.3.2.1). Here you can see that the largest topics found from the data are color-coded red, and subtopics are grey. Eventually the subtopics become so small they are defined as just words which are color coded blue. This kind of GUI gives users a sense of how prevalent that topic is comparatively to others.

The traversal mechanics are still the same as the people web, and users can traverse the graph by double clicking on nodes and also drag nodes by click and drag.

Figure 3.1.3.2.1: Example of the Topic Model graph

The look and feel give you a good idea of what’s related to what, and browsing through topics gives you a better feel of the subtopics. The overall aim of this GUI was to create an easily browse able and interactive environment. The user already gets a lot of information visually from the different node sizes, and thus we didn’t want to add the tool bars to affect node
size, thinking it would only clutter the feel of the web. The people web, because it was more simplified, allowed us to place more options to create a more interactive page.

3.1.4 External Apps

Since the external applications are separate and not embedded in the Pinquu webpage, there is a need to install them on the user’s local desktop. After installation, there are comprehensive steps to follow in section 3.2.4.

Instant Messaging

Install Spark 2.5.8 if you have a Windows/Linux machine or Adium 1.4.1 if you have a Mac machine. After user authentication at the user login window, users can start chatting with other users in their pre-populated contact list.

Document Sharing

After user authentication at the iFolder login page, users have the ability to create their own iFolder. Note that each user is allotted one iFolder. The specific steps to follow in maintaining each iFolder are detailed in section 3.2.4.2.

3.1.5 Mobile Workers

3.1.5.1 Virtual Whiteboard

The user can project the paint program on a wall and use the Virtual Whiteboard implementation to draw on the screen with IR pens. Multiple users can draw at the same time using up to four IR pens.

3.1.5.2 Status Stone

The user can physically flip the Status Stone to set his or her status to available/unavailable while concentrated in another task on the computer.

3.1.5.3 Mobile GUI Application

The user can log on to the Kiva database and surf through recent updates or browse the profile, calendar and spaces panels.

3.2 Usage Scenario (Illustrated by screen images and pictures of the equipment)

3.2.1 HCI

The HCI usage scenarios are described below in 3.2.2 Visualization/IOV Usage Scenario.

3.2.2 Visualization/IOV

Below in figure 3.2.2.1 is the user experience of a new user through each page. While HCI has described the overall user scenario in the earlier sections, we describe the scenario for an individual using the GUI exclusively with no other dependencies on hardware or communication patterns. This gives a better flow of the system usage for a person with GUI access alone without interruptions from other devices.

Harry Bovik, a user of Pinquu goes to the Login Page and types in his username and password.
Figure 3.2.2.1: Login page

Harry being relatively new to the company is still deciding projects that he wishes to pursue. A helpful feature for him is the news feed. Here he finds information on current projects at his company. Harry browses the list of updates on the news feed to get an idea of the people, projects and topics that interest him.

Figure 3.2.2.2: News Feed Page

While browsing through the Newsfeed, Harry finds this one project that matches his skills and interests. Eager to learn of what the project is about, who is in it and what they do, Harry clicks on the link on the Project page.

The first thing he sees is the Project Description. This summarizes the topic and current status of the project briefly. Seeing that the project is interesting and still active, Harry wants to read more about it. So he continues reading the documents uploaded by the members of the project team on the Project Page’s NewsFeed. Harry can also see the Top Contributors to the project and thus knows whom to contact if he wants to be involved in the project.
After reading about the project, Harry decides that he wants to be a part of that project’s team. However due to his interdisciplinary interests, Harry is not sure where he would fit into the project the best. So he decides to use the expert finder tool to see whether there exist experts in the specific field within the project he is interested in, or whether he would be very useful as one of the few experts. He types in the area of his expertise as the query in the Search box in the pane above.

Although he finds a list of projects that require expertise in that field, Harry sees that there are no people who are experts in the field. Harry now knows with certainty that he would be very helpful as part of the team.

Figure 3.2.2.3: Harry with the Project News Feed

Knowing that there is a need for him in the team, he decides to contact the manager of that team to inquire on its status. Harry types in the name of the manager of the project in the
search query (he had found the name of the manager on the Project page). This opens the search results page with the manager being in the results. Harry clicks on the manager’s profile.

Figure 3.2.2.5: Contact Information of the manager of the project

Harry emails the manager of the project through information he finds on the project page and asks if he can join the team. He also includes information on where he thinks he fits into the team and how he can contribute to the team based on the Project Information he has read on the Project Page.

3.2.3 Content Analysis

3.2.3.1 Scenario 1 – People

In this Scenario, Kendra wants to utilize the “View Similar People” function of the profile page to browse people of similar expertise. This way she can find coworkers to ask for help, or to more closely interact with peers who share similar interests.

Figure 3.2.3.1.1 a: Sample People web from Birdeye. b: Control bars on the side

Kendra thus opens up the People Web and she is brought to a web page with the generated flash file embedded into the page. In the center is Kendra and she can view the people and the various relationships she has with those people. What she sees is Figure 3.2.3.1.1 a with the control bar on the side as shown in Figure 3.2.3.1.1 b.

In the center is Kendra and when she clicks on the node name of people, more information is displayed on the top left corner of the Graph as shown in figure 3.2.1.2 (implemented only for Topic Model so far, but easily portable to people web).
As double clicks on nodes of other people, the graph shifts so that the newly selected node is now in the center. She browses for a while, clicking on people and reading their descriptions and expertise, seeing who she wants to contact with the most. Eventually after working her way through all the various people, she wants to go back to herself to see again the people most closely related to her expertise. However, she realizes that amongst all the browsing, she can’t seem to find her node in the web. So she decides to hit the refresh button on the side bar (Figure 3.2.3.1.1 b), and it resets the graph yet again with her in the middle. Now with the graph reset, she decides starts reviewing people closest to her and actually clicks and drags the users to the side that she finds interesting (Figure 3.2.3.1.3). She then browses through her selected nodes to find the optimal person to contact.

In the end, Kendra clicks on the name of the person that she most wanted to talk to which brings up the person information (Figure 3.2.3.1.2). Though not implemented, ideally there would be a link to that person’s profile page in the description. And she clicks on it to see the best way to get in touch with that person.

3.2.3.2 Scenario 2 – Topics

In this scenario Thomas is doing a project on sensors and needs a better sense of how sensors fit in with the rest of his class. He clicks on the “Wall of Projects” link on the homepage which opens up the Topic Model Web (Figure 3.2.3.2.1).
Similar to the People web, Thomas is able to traverse through the topics by double-clicking. Thomas is able to traverse. Also, as he clicks on nodes, in the top left corner appears some more information about the topic he is interested in (Figure 3.2.3.2.2). The Web also allows for him to drag nodes and rearrange, but upon seeing all the different subtopics, he also wanted to get a view in a smaller scope. He looks at the bottom left and realizes the slider bar which allows him to control the degrees of separation, and adjusts it to a comfortable level for himself (Figure 3.2.3.2.3).

In the end, Thomas is able to get a better sense of the research his class has been doing as a whole, and finds that one particular kind of sensor interests him. He clicks on that node bringing up the extra information (3.2.3.2.2), and ideally there is a link there that would redirect him to a page that has more information regarding that topic, like the search page for that topic (not implemented).

### External Apps

#### Instant Messaging

As mentioned in Section 2.3.6.1, our final choices of IM clients are Spark 2.5.8 for Windows/Linux platforms and Adium 1.4.1 for Mac OSX platforms. The user interface for both IM clients is intuitive and easy to use. In addition, administrators can manage the user database by logging into the Openfire 3.6.4 server using the admin account.
Server - Openfire 3.6.4

1. Log In to Admin Console

![Log In Page of Admin Console](image1)

**Figure 3.2.4.1.1 Log In Page of Admin Console**

An admin user is able to log in to the server that we have installed via IP address (http://128.2.83.208:9090) or via domain name (http://rpcs2011.andrew.cmu.edu:9090) from a remote desktop. This will bring the admin user to the Admin Console Log In page.

2. Server Settings in Admin Console

![Server Information](image2)

**Figure 3.2.4.1.2 Server Information**

After logging in, the admin user can view server information such as those illustrated in Figure 3.2.4.1.2. There is additional server information available under the “Server Manager” tab, as can be seen in the sidebar illustrated in Figure 3.2.4.1.3.
3. Managing Users/Groups in Admin Console

There are various other tabs such as “Server Settings” and “Media Services” that the admin user can view and edit accordingly.

Figure 3.2.4.1.3 Server Manager

The admin user can have a quick glance at the current users. By clicking on each user name, he can find out more information about that user, such as status and email address. Also, the admin user can create new users and grant admin rights to them. When the list of users becomes too long, the admin user can click on “User Search” to find a particular user.

Figure 3.2.4.1.4 Current Users

4. Sessions/Group Chat

Figure 3.2.4.1.5 Active Sessions
The admin user can monitor any active client sessions by clicking the “Sessions” tab. This also provides better knowledge of what other users are holding IM sessions.

Figure 3.2.4.1.6 Create Group Chat

In addition to monitoring individual chat sessions, the admin user can create group chats and monitor them too. This is easily done by filling up the “Create New Room” form.

IM Client – Spark 2.5.8 for Windows/Linux
Add an Account (can skip this step)

Figure 3.2.4.1.7 Spark: Add an Account

For Spark, there is no need to add an account for Kiva users since its integration with the Openfire server already populates their IM accounts. As such, Kiva users can skip this step.
Sign In

Figure 3.2.4.1.8 Spark: Sign In

In order to sign in to Spark, Kiva users enter their username, currently “StudentG” (can be any username in the current database) and the set password of 13 x-es: “xxxxxxxxxxxxx”. The server would be rpcs2011.andrew.cmu.edu, which differs a little from the sign in screen for Adium, in Fig. 3.2.4.1.11.

Contact List and Chat

Figure 3.2.4.1.9 Spark: Contact List and Chat Screens

After signing in, users are brought to the contact list, which shows other Kiva users in the database, split into their corresponding groups. All the groups in the Kiva database are shown in the contact list. To start a chat, users simply double click on a user’s name in the contact list. The contact list and chat box is relatively plain compared to that of Adium’s, in Fig. 3.2.4.1.12.

IM Client – Adium 1.4.1 for Mac OSX
Add an Account

Figure 3.2.4.1.10 Adium: Add an Account
Since Openfire was not meant to be directly interfaced with Adium, Kiva users would have to create a new account by following these steps:
- Adium > Preferences > Accounts
- Click “+” to add an account and select “Jabber” as the desired protocol.
This will bring you to the sign in page as illustrated by Fig. 3.2.4.1.11

Sign In

Figure 3.2.4.1.11 Adium: Sign In
Enter user information as seen above. The Jabber ID would be in this format: 
<username>@rpcs2011.andrew.cmu.edu, which differs slightly from that of Spark. The set password is the same for all the Kiva users in the database: 13 x-es, “xxxxxxxxxxxxx”.
Contact List and Chat

Figure 3.2.4.1.12 Adium: Contact List and Chat Screens

After signing in, users will see the contact list at the side, with all the Kiva users in the class pre-populated and categorized into their corresponding groups. Users can start chatting with other users on their contact list simply by double clicking on the user’s bubble. The appearance of both the contact list an the chat screen can be easily customized by following these steps:
- Adium > Preferences > Appearance
From that tab, users can change the appearance of their Adium.

Other common IM features
Both of the IM clients that we have chosen also support presence notification – letting users know when some other user is available to chat, busy, away or offline. Profile pictures can be customized as well.

Spark 2.5.8
- Modify general preferences by: Spark > Preferences
- Modify your profile by: Connect > Edit My Profile.
- Modify your contacts list by: Contacts
- Modify your status by: dropdown menu below nickname in contact list window
- Start group chat/conference by: Actions > Start a Conference
- Modify/Add plugins by: Connect > Plugins
- Presence notification (not as customizable as Adium): Spark > Preferences > Sounds

Adium 1.4.1
- Modify general preferences by: Adium > Preferences
- Modify your profile by: Adium > Preferences > Personal
- Modify your contacts list by: Contact
- Modify your contacts list display by: View
- Modify your status by: Status
- Start group chat/conference by: Contact > Initiate multiuser chat with <username>@<domain name>
- Presence notification: Adium > Preferences > Events
3.2.4.2 Document Sharing

iFolder’s intuitive user interface allows for easy maintenance of iFolder users, storage, and quotas. iFolder is run as a server that can be accessed from any location via the website rpcs2011.andrew.cmu.edu/admin or rpcs2011.andrew.cmu.edu/iFolder. The difference between the two websites is simply that /admin is the login page for administrative users and /iFolder is for all clients. Additionally, as a regular user of iFolder the clients have the ability to download the client software which handles effortless synchronization.

As a user, one is able to simply login using the credentials from the KIVA website. If this is a user’s first time logging into iFolder then they will use a default password given to them of “rpcs2011” and this can be changed at their discretion.

![Figure 3.2.4.2.1](image1)

Figure 3.2.4.2.1 shows the standard login screen.

![Figure 3.2.4.2.2](image2)

Figure 3.2.4.2.2

Once logged in the user has the ability to create their personal iFolder. This iFolder can be named anything the user desires but they are only allotted one iFolder per user. Within this file, the user has the ability to add files and even subdirectories. Refer to figure 3.2.4.2.2 for what the upload screen should look like. The folder poshinh is a subdirectory and the user Po Shin Huang can add files as desired. Additionally, the user can share these files with anybody else in the system directly.
The client removes the necessity of having to log into an online interface to manage documents. Figure 3.2.4.2.3 shows the client and using this a default folder is chosen within the local computer, which automatically synchronizes with iFolder. In this figure it is actively synchronizing and once it is completed it will say synchronized.

To further explore how to add a file, figure 3.2.4.2.4 shows that the file RPCS2011-03-18.txt has been added to the user Rapid Prototyping.
Figure 3.2.4.2.5 Evidence the text file has been copied over to iFolder. Once the client has been installed on the local computer, by simply going to the file that automatically synchronizes the file has automatically been added and is synchronized with user poshinh.

3.2.5 Mobile Workers

Figure 3.2.5.1 Using the Virtual Whiteboard
Figure 3.2.5.2 Status Stone Hardware – Available (left), Unavailable (right)
Figure 3.2.5.3 (left) Pinquu login page.  Figure 3.2.5.4 (right) Pinquu profile page.
3.2.6 Isolation Layer

The Isolation Layer is a backend tool that is not intended to be used by the users. However, the Isolation Layer is used by nearly every software application to ensure a consistent format to interact with the database.
4. Detailed Design

4.1 HCI

4.1.1 Functionality and Interactions

Pinquu’s basic functionality aims at allowing users in the office to better manage their own projects, keep a track of other colleague’s projects, and get help for any coding-based problems that they may face.

This functionality is achieved through various features found in Pinquu. The home screen allows the user to easily get an overview of the various projects that people are working on, search for specific people or files, and also view a news feed that involves status posts by the user’s colleagues.

The user can use Pinquu for getting an overview of the availability of other people intricate to his/her project as well as other colleagues in general. To further assist the user there exists a project visualization that provides an at a glance view of the various projects taking place and how they relate to each other.

One of the key features in Pinquu is that of finding an expert for a task the user might need some assistance with. The user can also chat with other colleagues and get any query he/she may have answered right away.

There also will be available a mobile version for Pinquu that would allow the user to view his/her projects, calendar, and the live feed through his/her android based mobile phone.

4.1.2 Screens

Homepage:

As soon as the user logs in to Pinquu, he is directed to the home page. Here the user has access to everything inside Pinquu. In other words this is the user’s dashboard.

1) On the left side, the user can see the projects he is currently involved in, and see when was the last time that something was updated to it. This is consistent through the entire web site.
2) The search bar is use to find anything in Pinquu. It contains filters to narrow down the search results to documents, people, or projects. This is also consistent throughout the entire web site.
3) The news feed is particular to this section of the website. The reason that is located here is so that the user can see what recent activity is occurring within Pinquu. Also, the user can easily upload a new file or post a new message to Pinquu.
4) In this section the user can access his profile by click on his name. Also, the user can log out from the system.
5) The calendar displays all the events and important deadlines related to the user. Every time a day is selected specific information to that day is displayed.
Search Results page:

Once the user has entered his search he is displayed with the following search results. In this case, the search result shows people, projects and documents related to the search keyword as the user has checked the “all” checkbox.

1) This is how the search results are displayed. In the top row it show the experts related to the search keyword. The next row shows the projects, which contain information about the related search keyword, and the last row shows the documents that contain information about the search keyword.

Project page:

This is a project page. Usually the user can find all the information related to a project in this section.

1) Once a project page is open the top collaborators of the project are displayed. The user can hover the pointer over any of the collaborators names to find out more information about them. If he wants to access the collaborators profiles or start chatting with them, he needs to click on the person’s name. Also the user can see whether or not a person is available to chat by the color of the rectangle.

2) Every project page contains a small description of the project being looked at. Also, if the project is not the one of the projects the user is currently working on, he has the option of following it. By doing this, the user can have get updates related to this project.

3) Documents, Messages and ‘To Do’ is the way the information in the project page is divided. In this screen, the documents are displayed. Documents are arranged within folders. The user can upload new files to a specific folder and create a new folder or document as well.
4) The button “View Wall of Projects” takes the user to a visualization of all the current projects in Pinquu.

Figure 4.1.2.2: Search Result

Profile page:
This is the profile page of any Pinquu user. Information about the person’s skills and experience is displayed here.
1) In this section the user can see the person’s online status as well as their contact information.
2) Indicates in which projects the person is currently involved with.
3) Information about the person’s skills and professional experience.
4) This button “View Similar People” launches a visualization about similar profiles to the one the user is currently viewing. The similarity is measured based on the person’s skills and professional experience.
5) All the documents and messages that this user has posted recently are shown in here.
Wall of Projects page:
This page can be accessed from the “View Wall of Projects” button in the project page. The main purpose of this visualization is to display how different projects relate to each other. Ideally, the user can indicate how he wants to search for similar projects. Maybe the user is interested in projects that have the same timeline, or require specific skills, or will take place in specific locations.

Whenever the user selects a project, it is moved to the center of the visualization and a brief description of the project is displayed. If the user is interested in this project, he can find out more information about it by clicking on the “View More” button.

4.1.3 Experimental Evaluation

4.1.3.1 Think Alouds
Think Alouds were performed with real users to determine breakdowns in the design of Pinquu. Users were given specific tasks to complete with Pinquu and the HCI team recorded what went wrong and what went right. The results from the think alouds are below divided by page.

Homepage without Search Text:
- The projects under "My Projects" should be listed with the project I have worked on more recently first.
- It would be helpful to bring back the link to the "Wall of projects" to the sidebar where "My projects" are and create an icon of the visualization
- A user never notices the check boxes for filtering search results.
- Didn’t realize that you can search forum discussions.
- Specifically asked if could because he didn’t know where to start.
- Naturally wanted to search for a person’s name - someone that he knows.
- It was confusing that documents, people and projects are selected when all is selected - suggests graying out the other options.
- Homepage with Search Text
- Most users tended to select only "people" when searching for an expert as oppose to selecting "all"
- The titles should be labeled "Documents in Pinquu network" and "Projects in Pinquu network" or something along those lines instead of only "Documents" and "Projects"
- Users want to look for some sort of directory of people

Search Results Page
- Unsure about the difference between experts – "why do I get these people when I search for experts?"
- Use existing chat conventions to indicate that a person is online. Using a circle or labeling the square with "available","unavailable","away",etc.
- Users want to click on the authors of the documents under "Document" section after they found out that the people listed on the search results are unavailable
- The red bar for indicating unavailability is not intuitive. A user suggests to put text on it.
- A user expects to see individual’s schedule for availability.
- A user wants to see project lead, dept., and team for each project in the search results.

Project Page
- Do not understand the language "wall of projects". Because it is located in this page it feels like is referring to the wall of a specific project, just like facebook uses wall under a person’s profile.
- Use "view all projects" instead of "wall of projects.
- Users felt uncomfortable browsing through project pages they were not a part of.
- "Documents" should be highlighted in grey and "Messages" and "To Dos" in orange (consistency)
- Use an arrow or some sort of indicator in the folder selected to indicate how the documents are related to the folders
- Users like the main layout of the project. It’s clear.
- Wouldn’t go here to look for an expert.

Project Page with People Profile Pop-up
- A user expects the contributor image to be clickable and goes to the profile page.
- Users like the overlay brief description when mouse over.

Employee Profile Page
- Users also want to see people’s extension numbers and office numbers.
- Wanted to see a list of forum discussion points - didn’t get this from the recent activity.
- Wanted to be able to browse previous answers to specific questions.
- Didn’t realize there was any IM component.
- Didn’t realize that the red bar meant unavailable.
- Once told it would say unavailable wasn’t sure what the unavailable would refer to - phone, email - what?
General Feedback
- Use orange color to indicate that something is clickable – keep consistency
- Use real project names and not "project Y", "project X"
- Projects in search result should not be the same as the projects under "My projects" area
- Having a breadcrumb would be useful to know where you are in the site
- 4.1.4.2 Heuristic Evaluations
- Two usability professionals performed separate initial Heuristic Evaluations in order to make sure all issues with the new system were addressed. Then a third usability expert reviewed and combined the evaluations to arrive at the final evaluation and score.
### Task: Find an Expert. Overall Average Score: 2.05 – Needs Improvement

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Rating</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visibility of system status</td>
<td>3</td>
<td>Pinquu keeps the user informed as to what is going on in several ways: real-time employee statuses; the mini feeds found on the home page, project pages, and employee pages; and the checkboxes by the search to note the type of data being returned. Also, for the most part, the browser will indicate system status for the user. The user sees system response if a new webpage loads. When contacting an expert, the user will know that the system has responded if, for example, the expert’s profile page loads. The user will also see an IM window when attempting to connect with an expert through chat.</td>
</tr>
<tr>
<td>Match between system and the real world</td>
<td>3</td>
<td>The vocabulary used in Pinquu is mostly user friendly. “Wall of projects” is a little abstract and doesn’t necessarily follow what employees call a group of projects in the workplace. Otherwise, everything is fairly easy to comprehend and follows real world conventions like red meaning unavailable and green meaning good to go.</td>
</tr>
<tr>
<td>User control and freedom</td>
<td>0</td>
<td>Couldn’t evaluate Pinquu on this standard since the demo only allows for certain tasks to be completed in a certain way. Ideally, Pinquu would support the user in every endeavor.</td>
</tr>
<tr>
<td>Error prevention</td>
<td>3</td>
<td>Appropriate feedback is essential for users to make better decisions. Pinquu includes the check mark boxes during the search to allow you modify your search result to more accurately find what you’re looking for.</td>
</tr>
<tr>
<td>Consistency and standards</td>
<td>3</td>
<td>Pinquu uses a consistent way to label the availability of users: green for available, red for unavailable. These colors are consistent with other IM clients in the market. The hierarchy of information is also displayed consistently in the same order.</td>
</tr>
<tr>
<td>Flexibility and efficiency of use</td>
<td>3</td>
<td>For employees who know exactly what they’re looking for, the customizable search with checkboxes aids expert users in finding exactly what they need.</td>
</tr>
<tr>
<td>Error prevention</td>
<td>3</td>
<td>Appropriate feedback is essential for users to make better decisions. Pinquu includes the check mark boxes during the search to allow you modify your search result to more accurately find what you’re looking for. Aside from error prevention on the side of interacting with web elements in the browser, Pinquu also prevents error of accidentally contacting an expert. That is, the user must go through several steps before contacting an expert.</td>
</tr>
<tr>
<td>Help users recognize, diagnose, and recover from errors</td>
<td>0</td>
<td>This feature was not flushed out for the demo and as such cannot be tested.</td>
</tr>
<tr>
<td>Recognition rather than recall</td>
<td>3</td>
<td>The minified includes several icons of the different activities to denote the type of activity it is. This helps the user quickly glance at the different activities rather than reading through each one.</td>
</tr>
</tbody>
</table>
Task: Find a Project
Overall Average Score: 2.1 – Needs Improvement

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Rating</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visibility of system status</td>
<td>3</td>
<td>Pinquu always heads its pages to let you know where you are in the website. Also, for the most part, the browser will indicate system status for the user. It is important for the project visualization to display proper visibility of reloading.</td>
</tr>
<tr>
<td>Match between system and the real world</td>
<td>3</td>
<td>The wall of projects uses proximity of nodes to denote similarity of projects, which is how people currently group items to show similarity.</td>
</tr>
<tr>
<td>User control and freedom</td>
<td>3</td>
<td>Users can search for projects through the search or with the wall of projects, giving the user a chance to search for a project in a variety of ways. As long as the website remains in non-flash form, the user has much flexibility built into the browser to go back or forward a page to undo an action.</td>
</tr>
<tr>
<td>Consistency and standards</td>
<td>2</td>
<td>The colors used for the wall of projects isn’t consistent with the rest of the coloring scheme of the website. However, Pinquu uses consistent visual elements and words to help the user find a new project. This includes placing the project name consistently at the same spot for each project page, for instance. We also use consistent icons to indicate the type of activity in the activity feed: message, announcement, document.</td>
</tr>
<tr>
<td>Error prevention</td>
<td>2.5</td>
<td>A short description about what the wall of projects is and how to use it would be more effective and prevent users from switching products by accident.</td>
</tr>
<tr>
<td>Flexibility and efficiency of use</td>
<td>2</td>
<td>For the expert, finding a new project would be fairly intuitive. They know exactly how to parse information from a project page to look at top contributors, documents, and general information about the project. Again, the expert could benefit greatly from keyboard shortcuts. Also, it would be better to allow expert users to access the wall of projects from their home page rather than through a project page. This would allow for faster interactions.</td>
</tr>
<tr>
<td>Aesthetic and minimalist design</td>
<td>2</td>
<td>While very clean like the rest of the site, the wall of projects could change its coloring scheme to match the rest of the website.</td>
</tr>
<tr>
<td>Help users recognize, diagnose, and recover from errors</td>
<td>0</td>
<td>This feature was not flushed out for the demo and as such cannot be tested.</td>
</tr>
<tr>
<td>Help and documentation</td>
<td>1</td>
<td>Besides the small blurb explaining what the wall of projects is at the top, there are no instructions about why it’s useful or how to navigate the interactive projects.</td>
</tr>
</tbody>
</table>

4.1.3.2 CogTool Analysis

Cog-Tool is a tool designed and created by a team at CMU led by Bonnie John. The tool uses the principals of KLM (key-stroke level model), which takes into account, for instance, research driven estimates on the amount of think time required and the distance between controls.
Cog-Tool is best used when comparing two interfaces for a single task. Since Pinquu is looking to support new interactions, cog tool could only be used to compare the task of looking for an expert in both interfaces.

We used the following task steps to compare the two interfaces:
- Find an expert that can help me solve my problem
- Login
- Search for specific coding problem in home page search field
- Look for a person or project to help with the problem
- Check the availability of the person
- Attempt to contact the first person found
- (Repeat until you can find a direct IM or phone number to call someone for help)

The following is a visualization comparing the 2 interfaces:

The overall finding for this task:
Pinquu time to complete: 28.9 sec
Kiva time to complete 69.9 sec

Conclusion: The Cog Tool analysis shows that Pinquu is twice as fast at completing the task of finding an expert as the Kiva website. For the Kiva estimate, this is the best-case scenario and assumes that one of the experts entered a phone number or the expert responds to an email. Being able to see the availability on IM in the website, even if the IM is separate, is a huge advantage and will make it much faster to find an expert in Pinquu than in Kiva. This is a large improvement upon the existing interface.
4.2 Visualization/IOV

4.2.1 Functionality

The role of Information Organization and Visualization is to implement the design Pinquu into the actual website. IOV is depended on many different groups including HCI GUI, Isolation Layer, and Content Analysis group. IOV is closely related to HCI GUI group as we need to come up with complete design that is possible to be implemented. Interaction with HCI GUI group was crucial as raw design sometimes go beyond some technical capabilities. For similar reason, there is strong connection between IOV and Database and Content Analysis group as functionalities are heavily depend on the different features that would be extracted from database. For example, information of experts has to be possible to extract from the database in order to implement expert finder page for Pinquu. On the other hand, IOV is giving out requirements to Isolation Layer as it will be the place where actual communication to database is handled. Therefore inputs and outputs of Pinquu websites would be given to Isolation Layer group to establish the connection between Pinquu website and the database.

Implementation of design was done in Visual Studio and all Pinquu pages are using CSS style in the front-end and written in ASP.NET and C# in the back-end design. XML is used to communicate with Isolation Layer as request of the data and retrieve of the data are all done in XML specification. Working closely with HCI GUI group, some changes in design were necessary due to some technical and time constraints as some of features cannot be done in given time. As a result, we carried out in total of seven standalone designs which are Login, Master, Home, Profile, Project, Search Result, and BirdEye Result Page. Here are brief functionalities of each page:

• Login Page: This is the entrance to Pinquu website where authentication is processed. Data is encrypted using ASP.NET and send to the database to verify user id and password. Access to the other page will always go through the login page.

• Master Page: This page is part of every other page except the login page. There are three components of the master page which are search bar (Top), project lists (Left) and calendar with meeting schedule (Right).

• Home Page: Main functionality of this page is to show news feed of the company that would help user to understand what is going on the company.

• Profile Page: In this page, user information including their name, title, area of expertise and their status are available.

• Project Page: This page will contain information you need to learn about project including its descriptions along with attachment of the project.

• Search Result Page: This page will be in result of search bar in the main page. There will be different categories that one can search for which are people, project and skill. In addition, there is a link to the BirdEye graph diagram. It would provide a complete view of topic one has searched for.

• BirdEye Result Page: In this page, there would be a flash embedded to the page which would show a graphical analysis of a search result. With the graph, user would be able to see how project or people are connected to each other.

4.2.2 Components

Information Organization and Visualization developed website, Pinquu, composed of 5 major pages: login page, main/news feed page, search page, project page, and profile page. Each page has different functionalities and shows different information in several formats.
Login page is the simplest page of all. User is shown a couple of textbox for input: user ID and password. If either password or user ID is incorrect, the user is not allowed to move to main page. Once ID and password pair is verified, user is automatically redirected to main page.

The main page, also known as “news feed” page, displays simplified but informative overall information regarding the user’s interest. The focus of this page is the middle area of the page, which displays recent activities that others did on projects that the user is participating. User is able to create an event or share a file. On the left side of the page, it shows projects that I am currently participating. On the right side of the page, it shows calendar and meeting notes to remind one. Also, there is a search functionality to search for documents, projects or people.

On the project page, top and right side of the page remains the same as the main page, but left side and main area are different. On main area, it shows any documents uploaded or comments made project participants for this specific project. This page is not limited to user’s own projects; user may view project page even if he is not a participant of the project. On the left side of the page, it displays top contributors for this project along with project description; therefore, anyone visiting this project information page may find the leader/expect for the project. Also, user can click on “wall of projects” to see project relational graph that shows closely related other projects that may be helpful.

On the profile page, top left and right side of the page remains the same as the main page; there are search functionality on top, my projects on left and schedule on right. The main area contains information of a person in one page. It includes person’s email, phone number, expert area, location and title. But more importantly, it displays the person’s recent activities of comments, events, or file sharing, as well as all the files shared on the bottom of the page. User can click on “view similar people” to see a relational graph of the person that shows closely related other people.

Lastly, the search result page displays people, projects and/or documents that are most closely related to the search keyword. Top, left and right side of the page remains the same as the main page. On search bar, there are three combo boxes: people, project, and document. User may check or uncheck the box to selectively search with keyword. Top 3 search results are shown on the page.
4.2.3 Experimental Measurements

Overall Design

Everything is designed to be as modular as possible so it is easy to remove things and plug things in. This also allows for easy hiding of certain controls should a user not have access to a specific control on a certain page. An example of this is the Top Contributors list in the left hand navigation. This list only exists on the Project Details page and is hidden if the user is on any other page. Certain pages also include ID data in the URL to display specific information about a selected item. This includes the Profile and Project Details page. In the URL, there may be a string of the form “id=3”, which specifies that the user wants to see Profile or Project information for a user or project with id “3”. This helps us to design the pages in such a way that it is abstracted from what the backend data is. The profile page is able to take in any user profile in the backend and display it, and the project details page can handle any project and display details of the project, as well as properly populating the Top Contributors list.

Along with the actual web pages, we also created user controls. The primary use for this is to eliminate duplicate code and to have access to simplified versions of existing web pages. An example of eliminating duplicate code is to have the recent activity GridView implemented in a user control. By implementing this once, the user control is then imported into the Project Details page and the Home Page with the news feed, and the recent activity for the corresponding pages is displayed properly with only one implementation of the GridView. User controls also allow for simplified versions of a user profile and project details. On some pages such as the Search Results page, we wanted to display a quick snapshot of information related to users or projects should a user search for them. This was accomplished using Simple User Profiles and Simple Project Details user controls. These user controls have a complete User or Project object bound to it, but only shows specific information in a condensed format, so many of these can exist on one page. Since they are implemented as a user control, we only needed to implement them once, and they can be data bound to any user or project in the backend, and literally hundreds of these could exist on a page, depending on the size of the search result.
Another method we used to reduce duplicate code is to introduce a CommonUtilities class, which houses commonly used methods and variables throughout the entire site. This includes string constants of keys which hash into the internal Session variable, string constants of web page names, and most importantly, the static methods to send and receive HTTP XML requests to and from the Isolation Layer.

**Custom Authentication**

Since we used an Isolation Layer on top of the database, we were not able to use the built-in authentication methods provided by ASP.NET. We created custom authentication to remedy this. We used the internal Session variable to accomplish this. Session is a private hashmap provided by ASP.NET, global and usable on all web pages. There exists an XML request to log a user into the system. We make an XML login request to the database. If it does not succeed, we will receive an error message and we set the Authenticated flag inside the Session variable to false. Otherwise, we will be returned with the UserId and a session token. We store these into the Session variable for later use and set the Authenticated flag inside the Session variable to true. Upon visiting every web page, there is a Page_Load method which runs as the page is being rendered on the screen. We check the authenticated flag here and if it is false, then the user is redirected to the login screen. Otherwise, the page is allowed to load for the user.

**Communication to Isolation Layer**

The procedure to send the request to the Isolation Layer is quite simple, but requires a strong foundation for the backend. There exists one C# file, which we will call the CommonUtilities, and it includes numerous static methods and variables to use throughout the program. Most importantly to the Isolation Layer Communication is a static method called GetData to send the HTTP XML request and receive a response. It is up to the calling code to properly create the XML request to send out, and to extract information out of the XML response that comes back. The calling code comes in three sections. The first section is to create the outgoing XML request, which is then passed to the GetData method in CommonUtilities. The second section is to verify that the response returned does not contain an error. If that check detects no errors, then the third section is to extract data from the response. The response varies from request to request. For example, a request for a user profile includes the user ID of the person to find, and the response returned would include first name, last name, email, expertise, etc.

In order to properly support the various XML requests and responses, custom C# objects to mimic the database objects were created. When the requests for a user profile from the database, it can temporarily store all the data inside a custom UserProfile C# object type until it is used.

The communication to the Isolation Layer can be further extended by introducing and implementing more XML requests that the Isolation Layer creates. The GetData method in CommonUtilities is complete and the backend foundation is clearly laid out: generate XML request and send to CommonUtilities, check XML response, and extract data from XML response. Inside each web page, and in some instances the user control, are methods to generate an XML request, check the XML response, and extract data from the XML response.
4.2.4 Software Architecture

In Phase 3, the responsibility of the Information Organization and Visualization group was to develop a graphical user interface for the web application, Pinquu. This meant designing seven pages – Login, Master, Home, Profile, Project, Search Results, and BirdEye Results. The web application was built in Visual Studio 2010 using C# in back end and ASP.NET for front end.

The pages have static information displayed, but also dynamic capabilities to retrieve information from Iso Layer to show things such as news feed, project documents, profile data, etc. Each page was customized to the specifications that the HCI group requested for. The
dynamic data was then parsed based on our own group’s specifications after XMLRequest queries were sent to the Isolation Layer.

4.2.5 Software Modules and Status

4.2.5.1 Implementation

The implementation of the design of the various software modules that went into the Pinquu web interface spanned four main sections:

- Site Master
- Home Page
- Profile Page
- Search Page

In addition to these, an implementation was created for the Login interface, but was mostly trivial, following the typical design pattern used by most web interfaces for authentication (see figure 4.2.5.1).
4.2.5.2 Site Master

The Site Master page encompasses the top navigation bar (4.2.5.2a), as well as the left and right navigational elements of the Pinquu web interface (4.2.5.2b and 4.2.5.2c). These three elements make up the Site Master page, which is in turn shown on every page of the Pinquu interface as a unifying navigational element. This was achieved through the use of an Asp.NET Master page, which could be treated separately from content pages that would be embedded within the master navigation page, as in Figure 4.2.5.3.

Figure 4.2.5.2: a) Top Navigation, b) Left Navigation, c) Right Navigation

Figure 4.2.5.4.3: Example page embedded within Site Master Page
Top Navigation

The top navigational module is comprised of three primary components: the logo and home link, the search area, and the account management area. These were treated as separate CSS <div>'s in practice, with precise measurements for width and height to maintain design unity. The search area was given a CSS gradient treatment in order to emphasize the importance of search as a means of traversing and utilizing the informational capabilities of Pinquu. The gradient makes the search bar appear to extend outward from the interface (Figure 4.2.5.4).

![Search Area](image.png)

**Figure 4.2.5.4: Search Area**

This code in CSS was designed specifically with cross-browser abilities in mind. Since much of the business world uses Internet Explorer down to version 6, care was taken that these CSS effects would translate down to older web browsers, as seen in Code Sample 5.2.5.1. The right side of the top navigation bar reveals the account management area, with the ability to log out of one's account, or click a link that takes the user to his or her own account profile.

```
linear-gradient(#fff, #f5f1ec);
/* for IE */
filter: progid:DXImageTransform.Microsoft.gradient(
    startColorstr=#ffffff, endColorstr=#f5f1ec);
/* for webkit browsers */
background: -webkit-gradient(linear, left top, left bottom,
    from(#fff), to(#f5f1ec));
/* for firefox 3.6+ */
background: -moz-linear-gradient(top, #fff, #f5f1ec);
```

**Code Sample 4.2.5.6: Gradient CSS**

Left Navigation

The left navigation in Pinquu is dedicated to information about the host company's projects. At the top of the left navigation pane, there is a list of the projects that the currently logged in user is a member of. This is a quick way for the user to access any of his or her projects and also see quick facts about them, such as when they were last updated with new content. Beneath the list of projects, there is a link to the Wall of Projects visualization, which is run through Birdeye and discussed in another section of this report. This area allows the user to visualize a broader range of projects that may project beyond his or her own, but still be relevant to his or her needs. A screen capture of the left navigation bar can be seen in Figure 4.2.5.2b.

Right Navigation

The right navigation bar is the area where the logged in user can see all of his or her appointments in either a calendar or a short list of upcoming events. The calendar is a Javascript widget available for use with Asp.NET projects in Visual Studio. The widget has been heavily customized for look and feel for Pinquu. Though the largest amount of customization possible was achieved through CSS, some of the Javascript code had to be directly manipulated due to its
tendency to override the CSS. The look of the calendar in its final form keeps relative fidelity to the design prototype proposed, with all usability enhancements in place. These include a distinct visual style for the current day, a selected day, the days of the current month versus adjacent months, and a simple way to cycle through various months of the year. Additionally, the event list below this calendar gives further details, such as time and location for the given event. See Figure 4.2.5.4.2c for a screen capture of the right navigation panel.

### 4.2.5.3 Home Page

The home page is designed to be a quick stop for users to see recent updates to any projects in the company they wish to see, and to share files or create events on the fly. The primary modules on this site include the News Feed and the Contribution area. The News Feed module (Figure 4.2.5.3.6a) shows recent activity from any selected project, including (and able to be filtered among) files, calendar events, and messages. The Contribution module (Figure 4.2.5.3.6b) allows a user to quickly post a file, event, or message to a project of his or her choosing. Like the search bar above it, the main content area in which the Home Page resides is styled with a cross-browser CSS gradient implementation that helps the content be emphasized to the user.

![News Feed and Contribution Area](image)

**Figure 4.2.5.3.6: a) News Feed, b) Contribution Area**

Due to time constraints, ability to load data from the News Feed or post to the Contribution Area is not working at this point.

### 4.2.5.4 Profile Page

The profile page reiterates some of the same functionality as the home page but specifically for the viewed user’s profile. For example, there is a "News Feed"-like area for Recent Activity for just that user. Additionally, the user’s current projects, skills, personal information, and uploaded documents are shown. A link next to the user’s name reveals a visualization to show similar people, while a colored bar underneath the name indicates the user’s instant messaging availability status. Most of the features can be seen in Figure 4.2.5.4.7.
4.2.5.5 Search Page

The search page is split into three modules: People, Projects, and Documents. Search results are split up to find people with relevant background or relevance for a given search, Projects with relevant topics, and documents with relevant titles and content. These are split into three tables for the users' ease of seeing top results for each separately. These can be seen (though with only the latter two categories having results) in Figure 4.2.5.8.
4.2.6 Future Work

Master page
- Left side bar (Current user’s project description). When user clicks on one of the projects, it expands showing top contributors and project description. According to the Original mock up design, user should see a brief description about each contributor when he/she puts mouse pointer over the name of a person. There are no XML specs for this.
- Right side bar (meetings and agenda for the user). Currently, there are no XML specs for retrieving meetings information from ISO layer. Information for meeting is currently hard coded. Future implementation needed so that user can view his/her schedule for any day in the calendar.

Home page
- Posting threads, uploading documents still needs to be implemented. Functionality where people can directly communicate within a post by commenting required.

Profile page
- There should be links to each expertise so that user can be redirected to the search result page showing related people, projects, and documents for that specific expertise.
- When user clicks on “View similar people” button, the BirdEye flash should load with the current profile page person centered in the middle. Now it just loads with default setting.

Project page
- At this point, there are missing XML specifications for getting information of a specific project, so only related documents are shown on this page for now.
- This page should be more dynamic so that user can upload/download files directly through this page. The best way to do this would be to find a way to embed iFolder so that user does not have to log in to iFolder in a separate window.

Search result page
- Find a better searching algorithm. Now this page first gets information of all the people within this company and sorts them according to the search text, but this method seems to be really slow. Same for the projects result part.

4.3 Content Analysis

4.3.1 Functionality

Our Team works with the concept of Text Extraction - specifically Topic Modeling to generate a list of topics for a given set of documents; kiva posts and any other information users may have uploaded. These topics are then visually represented using BirdEye.

In the ideal scenario, someone generates a topic model for a specific request each night based on the requirements of the company. However, this is what actually happens in a sequential manner on the backend:
- A script made by the Content Analysis group is run to automate the process.
- The script runs our code, which communicates with the Isolation Layer to get documents, posts, etc.
- The script then copies all the files (which are on the work machine) into one specific resources directory
- These files, irrespective of whether they are .ppt, .doc, or pdfs are then converted into .txt files and put in another directory called resources-text
- LingPipe is run across this resources-text directory to generate Topics, edges to other topics, sub-topics and so on
- Birdeye then uses these topics to create a visually appealing topic graph that the user can interact with.
- It is this topic model that is integrated into the Pinquu website.

In a similar manner, we are creating an Expertise graph where different people with the same expertise are connected to each other.

### 4.3.2 Components

The Database and Content Analysis team works with these different components:

**Isolation** – Content Analysis Port: This is something that we made to interact with the Isolation Layer. We send the Isolation Layer a XML request for a list of topics and the Isolation Layer responds with an XML response which contains the paths to where the different files for analysis are stored.

**Automation Script** - Automates the entire process. As our group is working with multiple individual components, we wrote a script, which automates the entire process. The user merely has to run this script to generate the Graph which would be displayed on the organization’s website.

**Lingpipe** - A toolkit for processing text using computational linguistics. By modifying our existing version of LingPipe we are able to create topics from uploaded documents, posts and other information that users may have uploaded. By working closely with the Isolation Layer, we are able to generate topics in different levels of granularity such as topics from a particular User, topics from a particular group and so on.

**BirdEye** – It is a community project to advance the design and development of a comprehensive open-source information visualization and visual analytics library for Adobe Flex. The actionscript-based library enables users to create multi-dimensional data visualization interfaces for the analysis and presentation of information. Through Birdeye, we are able to represent the topics generated in a visually appealing and intractable manner. From the Kiva database, we are also able to get information regarding the expertise of different users and generate an Expertise graph. Essentially, when two people have the same expertise, a link is drawn between them.

**Expertise Model Generator** – This was something we developed by bypassing the entire LingPipe Component and using Birdeye’s visualization abilities. We were able to get Expertise information of different users. We then used this information to create a graph through Birdeye where different users were linked if they had the same expertise.

**Topics Filter** – As LingPipe works through the use of various algorithms such as Latent Dirichlet Allocation (LDA), a lot of noise is created in the form of words such as ‘you’, ‘although’ etc. To solve this process, we added words to a textfile ‘stopwords.txt’. Essentially, words in this text file are not added to the topic model even if LingPipe determines that a word contained in stopwords.txt is a topic.

By using these components effectively, we are able to create a visually appealing and intractable graph of topics which both managers and employers could effectively use to find different people of a similar expertise, figure out resources need to be allocated to a new issue that may be cropping up and so on.

### 4.3.3 Experimental Measurements

Currently, our code is set to extracts topics from the list of files that it is given. A wide range of files is supported, include Microsoft Word, Microsoft Powerpoint and Adobe PDFs; the text from these files is extracted into a text file before being processed.
In the ideal scenario, the automation script copies over all the files to a specific directory, from which LingPipe can use to extract topics. We retrieved a list of all files continued in the data base as well as their location on the file system. However, privacy issues prevented access to the actual documents so we instead chose a dummy sample of files, all regarding sensors, to process.

To test out LingPipe and for the purpose of the demo, we fed it 6 different files related to sensors. LingPipe went to generate words like ‘environment’, ‘voltage’ and so on – all words which are associated with sensors. Even though the heading of many of these files were things like ‘The Future of Sensors’, ‘Future’ was not included as a topic because it was not a relevant topic according to Latent Dirichlet Allocation (LDA) algorithm. Sample topic models can be seen in Figures 4.3.3.1 and 4.3.3.2.

Figure 4.3.3.1 A sample topic web for several documents related to sensors.

Although we added some words to the filter we made, there was still some noise that was created because we did not add a robust list of words. We believe that this file has to be modified according to the needs of the user.
Figure 4.3.3.2 The same sample topic web with a subtopic ("sensor") at the center.

Each topic that was generated also contains various sub-topics that in turn have additional sub-topics linked to it. In the above example, the major topics such as ‘sensors’ have sub-topics ‘applied’ and ‘voltage’ associated with it. These sub-topics have other sub-topics associated with it.

When we talked to Dong Nguyen, a PHD student in the Language Technology Institute regarding improving the overall performance of LingPipe, she stressed the importance of having a lot of files to analyze. Her recommendation was to have at 100 different files to derive meaningful information from it.

We believe that LingPipe, even with its flaws, could prove as a powerful tool for ‘The workforce of tomorrow’. However, further development needs to be done to further refine the product. Specifically, tests on actual document sets need to be conducted to determine if Lingpipe is capable of extracting useful information for future workers.

4.3.4 Software Architecture

Figure 4.3.4.1 shows a visual representation of the software architecture used by the Content Analysis team.
Figure 4.3.4.1 Content Analysis Software Architecture

All software was controlled by an automated shell script and each box represents a separate computer program. To build a topic web, the script beings by calling our Java program that interfaces with the Isolation Layer requesting a list of files and their locations. Once this request is fulfilled, the automated script then extracts the text from these files and passes this list of files to Lingpipe. Lingpipe uses its natural language processing algorithms to generate an XML list of topics which are then passed to Birdseye which creates the final Flash file.

Creating a People Web is quite similar. The only difference is the data formatting and the fact that Lingpipe is not used.

4.3.5 Software Modules and Status

All of the boxes in Figure 4.3.4.1 are currently functioning. However, there is one issue with the Isolation Layer throwing an error when a list of all files is requested.

4.4 External Applications

4.4.1 Functionality

4.4.1.1 Instant Messaging

Server – Openfire 3.6.4

There are four main tabs on the server configuration window, which is administered via a web browser.

Server:

This is where the administration of the main server functionalities and properties are carried out. It is possible to update server properties through this tab rather than using MySQL commands but it is very cumbersome. Lastly, it also provides a quick access to the database through the database viewer. This was done by installing a “DB Access” plugin.

User/Groups:

This tab configures the profile and administers the different users and groups present in the network. Since we are utilizing a dual database configuration (refer to the software architecture in section 4.4.4.1) where user data is pulled from another database, the functionality of this tab has been disabled whereby you can only view the users but not edit them. It also
allows the quick importing and exporting of user data in certain beneficial cases such as server migration or server backup.

**Sessions**

Sessions show the different users currently connected to the IM network and other beneficial information such as status and IP addresses.

**Group Chat**

Group chat allows the administrator to edit settings for chat conferences or if need be, also create conferences manually and broadcast it out to the users connected to the network.

**Plugins**

Allows for the uploading of plugins to extend functionality real time without a server restart, which would disrupt other users connected to the network.

**IM Clients – Spark 2.5.8 and Adium 1.4.1**

**Login:**

Both IM clients require a username, password and server name in order to authenticate the users that are signing in. User authentication allows for the correct list of contacts (Kiva users in the class) to be pre-populated for each user, by pulling data from the Kiva database via the Isolation Layer.

**Contact List:**

This list is pre-populated based on the information in the Kiva database. Only the administrator managing the Openfire server and/or the Kiva database has the power to change the contact list as well as the groupings/projects that each user is a part of. The list is comprehensive in the sense that it can be sorted alphabetically or based on status. Users in two groups or more will appear in each group that they are a part of – other users can find them via their group names on the contact list.

**Start One-to-one Chat:**

For both IM clients, users can start chatting with other users by double clicking on their corresponding nicknames in the pre-populated contact list.

**Start Group Chat/Conference:**

Both IM clients allow for the initialization of group chats or conferences. This is detailed in section 3.2.4.1 of the report.

**Appearance – Profile Picture and Status:**

Both IM clients provide the functionality of changing users’ personal appearances, be it changing their profile pictures or statuses. This is detailed in section 3.2.4.1 of the report.

**Presence Notification:**

Presence notification can be pre-set in user’s preferences. Users can set their IM clients to have a pop-up notification/produce a pre-set sound when users sign in or log out or change their statuses. Steps to do so are detailed in section 3.2.4.1 of the report. It is nice to note that Adium’s presence notification is far more customizable than that of Spark.
4.4.1.2 Document Sharing

![Diagram of Document Sharing](image)

**Figure 4.4.1.2.1**

iFolder has the ability to share files between any number of different platform computers. The management of the users and individual iFolders is effortless using the online interface; however, the integration with Kiva takes a little more work.

As shown in figure 4.4.2.2.1, iFolder interacts directly with the clients but indirectly with the Kiva database through the dispatcher. There are two key scripts that have been written to address the interaction between Kiva and iFolder. The two interactions that needed to be addressed were:
1. File-paths
2. Accounts

File-paths need to be generated for each user displaying where their information is being saved by iFolder. The file-path has to be significant to the user. This done by several bash scripts that log onto the ASP interface of iFolder and for each user pulls the file-path for where the documents are being stored. This information is then put into an XML format ready to be sent back to the dispatcher.

Accounts need to be generated by using the XML dispatcher as well. It is easy enough to use however, parsing the XML is a little bit more difficult in Bash. To do this, a bash script that pulls all of the users from Kiva and, one at a time, uses a command line tool to add them to iFolder. If a user already exists then, the account will not be recreated. Thus to keep the list of users up to date, this script can be set as a cron job to cycle through all users. Or, the script can be used every time a user is added.

4.4.2 Components

4.4.2.1 Instant Messaging

Since IM is a software application, there are no hardware components involved. We are, however, making use of the desktop in the lab that has been installed with Linux OpenSUSE 11.2. In addition, we have installed Openfire 3.6.4 on it.

On client computers, we will be using Spark/Adium to connect with the Openfire server.
4.4.2.2 Document Sharing

The main server we have set up for iFolder is a Linux openSUSE 11.2 server equipped with large capacity hard drive and stable network. We opted for openSUSE because that is the recommended operating system to be used with iFolder. The client computers can be any computer: Windows/Mac/Linux PC, as long as it has access to a stable network.

4.4.3 Experimental Measurements

4.4.3.1 Instant Messaging

Maximum Number of Users

There is no theoretical maximum to the number of users as it is all dependent on the hardware specifications of our server, our internet speed and the number of active connections we have to our database.

Simultaneous Users

Again, it depends on our system resources and hence how good our server specifications and internet connection speed would allow us to determine the cache size we can allocate to maintaining the number of simultaneous users. So far due to the restrictions and limited scope of testing we have done, we have no problems supporting 10 users simultaneously on the test server allocated to us. We expect the number to be several thousands of users since the server supports clustering which enables us to deploy our solution across several terminals if we have a proper setup. A trial conducted by the developers of the software showed results of supporting 50,000 concurrent users at any particular time.

Message Delivery Speed

The message delivery speed was almost instantaneous but that is expected considering the proximity of our users. Results may vary significantly if the users are based in different locations or even countries.

File Transfer Speed

The file transfer speed is mainly dependent on the server memory and the internet speed. Thus results could be subjective and varies with individual client computers. For our purpose of testing, we sent a file of 100Kb between 2 computers and the transfer took around 10 seconds to complete which is a reasonable result considering our hardware specifications and internet connection speed.

Chat Conferences

It has been tested with our server that we could support a multi user chat or chat conferences of 20 users in a single chat room. It is possible to increase the number of users in a chat conference but that would depend on the performance of our hardware. As to how many concurrent chat conferences we could support, that would be left for future testing since we do not have the resources to test that at the moment.

4.4.3.2 Document Sharing

In order to gauge the capabilities and performance of iFolder, a set of metrics were tested and compared to other file sharing tools. Dropbox is used to represent a more established document sharing tool with a paid service and OpenKM is used to represent another open source document sharing tool. The following is a table showing the values gathered for each metric using each file sharing software:

<table>
<thead>
<tr>
<th>Software Used</th>
<th>iFolder</th>
<th>OpenKM</th>
<th>Dropbox</th>
</tr>
</thead>
</table>
Table 4.4.3.2.1 Document Sharing Software Comparison.

ifolder’s upload and download times are very reasonable and match the standard users would expect to see. Actually, ifolder performs the fastest in all of the tested cases. I believe this is due to the low user base on our ifolder server, which is extremely small in comparison to the user base on more established document sharing tools. Also, only a small percentage of our full server quota is filled allowing for higher speeds, whereas other tools most likely have a much higher percentage of their space full. When comparing the maximum possible disk quota space per user, ifolder and the other open source tools tend to beat out the more established tools. The length of the upload and download times of ifolder could probably be shortened even further if the server’s default storage changes from the local drive to the larger hard drive. This could be something improved upon in future work.

4.4.4 Software Architecture
4.4.4.1 Instant Messaging

Figure 4.4.4.1.1 IM’s Software Architecture

4.4.4.2 Document Sharing
4.4.5 Software Modules and Status

4.4.5.1 Instant Messaging

In order for us to proceed in this phase, we had to decide the software we needed to utilize subject to certain constraints such as cost and compatibility issues. The broad framework of our design could be seen in the previous section where we used OpenSuse 11.2, MYSQL 5.1, Openfire 3.6.4 and Pidgin 2.7.

OpenSuse 11.2

We chose OpenSuse 11.2 because we needed a Linux working environment for the iFolder portion of our project and OpenSuse was the recommended option by the developers of the iFolder project. For the IM portion of the project, the type of Linux distribution and version was not critical and did not require any special working environments. We also expect future stages and testing to be using the same OS as we do not have any major bug issues that require us to upgrade to a newer version.

MYSQL 5.1

In order to test our IM server, we needed a test database where we could store the user details and information. MYSQL was the no brainer option since the isolation group which hosts the KIVA database also uses MYSQL to maintain their database. The specification manual mentioned that any version of MYSQL later than 4.9 could be used. So far after our testing and configuration, our test database syncs very well with our IM server and we do not foresee any issues when we need to test our IM server on the actual KIVA database itself through the Isolation Layer.

Openfire 3.6.4

Initially, we tried the latest version of Openfire but ran into setup problems and hence we downgraded to the previous version of Openfire which was 3.6.4 and the configuration of the

Figure 4.4.4.2.1 iFolder’s Software Architecture
server went smoothly after that. The server has been running stably through our testing and we rarely had any server downtime issues after we had iron out all the bugs in our initial setup phase.

**Spark 2.5.8**

Spark 2.5.8 was chosen as the IM client for Windows/Linux platforms because it was the latest stable release and the fact that it synergizes very well with our Openfire server in terms of harnessing its plugins and ease of usage for the users appealed to us. With the dual database configuration, it is able to retrieve all the users and their groups associations from the database correctly and automatically with no administrator intervention required.

**Adium 1.4.1**

For the Mac OSX platforms, we decided on Adium instead since development of Spark for the Mac OSX platform was not very well supported and is lagging behind the other two platforms in terms of user interface features. Likewise, with the dual database configuration, it is also able to retrieve all the users and their groups associations from the database correctly and automatically with no administrator intervention required.

**4.4.5.2 Document Sharing**

The previous phase involved setting up iFolder on openSUSE 11.2 and identifying and installing all dependencies. A great portion of our time in the previous phase was spent on determining the right configuration settings for iFolder and its dependencies (Mono, openSUSE, apache).

In this phase, we focused on connecting iFolder to the database through the Isolation Layer so that information could be shared between the two. One task was to link user authentication so that a user could use the same username on both Pinquu and iFolder. The second task was to retrieve the paths to the files for each user, so that the database could locate the source of each file.

In order to automate both tasks, several bash scripts were written. One bash script sends an xml request to the Isolation Layer to retrieve the username information and creates a new user in iFolder through the UserCmd.exe tool.

The next set of bash scripts automates the task of sending file paths to the database. In order to keep track of all the files that each user has, only one iFolder is allowed per user. This way, only one iFolder needs to be searched to find all the files a user owns. The bash scripts successfully authenticate and log into the iFolder admin site and save a cookie to allow access to further authenticated pages. From the authenticated pages, a list of all current iFolders is retrieved. For each iFolder, the owner’s username and the iFolder’s local file path is retrieved as well. By using the “ls” command in linux, last modified time stamps and exact paths are retrieved for all of the files (some files may be in further subfolders within an iFolder).

Since the xml spec requires user requests to be made by user id, a bash script was written to map usernames to user id’s based on the information retrieved from the user creation script. This mapping is currently used in the xml request to set an iFolder path for a user, but it can be used in any future user xml requests.

Next steps include automating updates between the 2 systems. For example, if a user is added to Pinquu, an iFolder user should automatically be added. If a file is added to an iFolder, the Pinquu list of files should automatically updated. The changes made can be seen if the scripts are run, but someone has to manually run the scripts. For future work, it would make more sense for the Isolation Layer to handle the comparison between previous and current folder states. This was not feasible by a bash script because of the lack of data structures available.
4.4.6 Design Alternative Evaluation Chart

The IM team changed our choice of IM client from Pidgin to Spark for Windows/Linux platforms and Adium for Mac OSX platforms. We include an evaluation chart to summarize and justify the reasons for our final design choice.

The last two desired features were very important to us since the visionary scenarios wanted us to be able to group the users into projects/groups. Pidgin did not allow for real time updates since it had a hidden cache that wouldn’t reset the groups until we logged out and logged in. Even so, that was not guaranteed.

<table>
<thead>
<tr>
<th>Desired Features</th>
<th>Spark</th>
<th>Adium</th>
<th>Pidgin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chatting</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Group Chats/Conferences</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>File Transfer</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Presence Notification</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Availability Changing</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Authentication with Kiva Database</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Synching Friends/Groups with Database</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Real Time Update and Display of additional information</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 4.4.6.1 IM Feature Matrix.

4.4.7 Future Work

4.4.7.1 Instant Messaging

- Look into Status synching for IM
- Maybe an embedded version into Pinquu for IM
- Desktop sharing via the IM itself
- Ability to broadcast user status is governed via the IM client side and cannot be updated via the server database. Hence a direct connection to the status stone to update the client status has to be created via a listener software.
- Implement server clustering to reduce loads on the main server.

4.4.7.2 Document Sharing

- Automatically synchronize the user information and file path.
- Synchronize the file path according to the exact file system instead only reflect the new files.
- Synchronize the exact user database instead of trial and error on user IDs.

4.5 Mobile Worker

4.5.1 Functionality

4.5.1.1 Virtual Whiteboard Functionality

In today’s increasingly mobile workplace, group meetings and conferences are less and less likely to take place in a singular place and more importantly perhaps not at an office at all. It is important for the mobile worker to be able to give and attend presentations no matter where they are and also take that information with them once the meeting is over.

With these ideas in mind, we wanted to create a virtual whiteboard. This would be a whiteboard that retains all the flexibility of working with physical hardware, but introduces the
portability and storage capability of a digital system. The virtual whiteboard combines a projector (pico or regular), computer, and Infrared L.E.D. pens to simulate the behavior of a regular whiteboard on any surface in any location. Currently, the Mobile Worker team has written custom software based on Johnny Lee’s Multigrid program that allows up to four users to interact on a paint program simultaneously. A save function is provided that takes a screenshot of the current paint screen and saves the image in a PNG file to the desktop.

### 4.5.1.2 Pico Projector

The Pico Projector is a laser based portable projector, capable of being connected to a device with a VGA port or an iPhone, iPod, or iPad out of the box. The Pico Projector is always in focus due to the laser technology. The use case for this is to allow for a mobile presentation or conference. We have successfully mimicked a mobile user by connecting the Pico Projector to an iPhone 4 and was able to project image slideshows and videos. The next steps are to get a hold of the software development kit for the Pico Projector to increase the brightness, which should make it a more viable option for mobile presentations for a group.

### 4.5.1.3 Status Stone

A familiar use case for software developers is the need to quickly change one’s externally visible status without having to navigate to the status managing application. The status stone is a hardware solution for this use case; essentially, the status stone will be an external device which projects your status (as viewed externally) to you through the use of some LED lighting scheme. In the event that you want to change your status, you simply rotate the status stone and your externally visible status should transition from ‘AVAILABLE’ to ‘NOT AVAILABLE’ or vice versa. Furthermore, the LED lighting scheme should reflect what your externally visible status is so that you, as the user, are not projecting an incorrect status (i.e. showing ‘AVAILABLE’ when you are actually ‘NOT AVAILABLE’ or vice versa).

### 4.5.1.4 Mobile GUI Application

The mobile Pinquu application on the Android OS provides a way for teammates to stay connected to the project and their teammates remotely, while travelling. It will have Profile page where user can see the status message, set new statuses, and access their project pages. The project page will have the basic information along with the last item people posted in regard to the particular project. It will also have a Calendar page where user can see the upcoming 5 events. In the Spaces, user can get a list of 4 documents updated, and this list can be filtered so it only displays the last 4 documents updated by a certain author, or certain file type. All 3 lists--project posts, calendar events, and document list have a link at the end that takes them to the full website with a longer list.

### 4.5.2 Components and pictures

The Mobile Workers primarily works with the tablet laptop assigned to us and will be porting the Virtual Whiteboard to the desktop in the future. We used the M410HD projector as the projection source for the Virtual Whiteboard.
Figure 4.5.2.A Laptop & Desktop

The Virtual Whiteboard also required the use of a Wiimote and infrared (IR) pens. The Wiimote is a standard, prebuilt device from Nintendo and has a 45 degree cone of reception at its tip. The IR pens were created using a momentary button and an IR light emitting diode (LED) powered by two AA batteries. We also used a microphone stand to support the Wiimote and raised it as high as we could to eliminate interference in a room. A video of the Virtual Whiteboard use in practice can be seen at http://www.andrew.cmu.edu/user/rantaoc/

Figure 4.5.2.B Projector

The Pico Projector is a portable projector which utilizes laser technology. It can be connected to an iPod/iPhone via a custom cable, it can connect to any device utilizing composite video, and it also has a VGA dock.

Figure 4.5.2.C Wiimote, IR Pens, Wiimote stand
There were a couple of factors that would clearly need to be taken into consideration when we started to think about design of the IR pens. First and foremost, we wanted to be able to get a number of pens developed and out to different members of the Mobile workers team e.g. the pens had to be easily produced. This was essential to get testing of the smart whiteboard underway and would enable parallel development down the road. The second factor to which the solution wasn’t immediately clear was what form factor of the pen would prove to be both most ergonomic and allow for optimum recognition by the IR pens. With these two things in mind we set out to develop 3 prototypes.

Our 1st IR pen prototype was made not for durability but rather quick experimentation. It is simply all the electrical components wire wrapped and taped together. This pen was never meant to make it into the final production stages of our smart whiteboard module. We used this
as a staging ground to test the different battery and resistor setups that would be necessary to make the IR pen possible.

Figure 4.5.2.F 2nd Pen Prototype

The 2nd prototype was made to more closely reflect a real life dry erase pen that you would typically use on a standard whiteboard. Once chosen all the components where fitted into an old gutted expo marker. A hole is then drilled into the top of the pen to accommodate the push button. All connections are routed out the back and to the outside of the pen to make it easier to debug.
Figure 4.5.2.G 3rd Pen Prototype

The 3rd prototype is more of an iteration of the second pen. We replace the bulky expo marker with a cheaply replaceable body. In this case we chose to go with thick disposable straws. This not only cuts down on the amount of time it takes to fit the button but is also a cheaper alternative to the version 2 of the pen. Two of these prototype models were made.

A few lessons learned in developing these prototypes. First, is we are unhappy with both the size and quality of the push button. The quality of the buttons is inconsistent, providing inconsistent behaviors in terms of the push pressure required to activate the IR LED. Also the size of the button is a major frustration in terms of pen construction. There are not many ergonomic tubes that will fit the size of the entire button enclosure. We are currently looking to procure other models. Another important lesson is that we need to design somewhat of a firmer hold for the IR LED itself. This was not immediately apparent but in testing, different users use the pen at different pressures. When used more like a real marker, this sometimes caused the IR LED to retract into the pen thereby shielding it from the Wiimote’s IR camera.

Figure 4.5.2.H Status Stone Hardware – Available (left), Unavailable (right)
4.5.3 Hardware Architecture

The mobile GUI is designed to currently only run on Android OS, and its look and feel has been specifically optimized for the Google G1 developer’s phone with 320x480pixel screen resolution. Its main input is via the virtual keyboard and the touch screen.

4.5.4 Software Architecture

Figure 4.5.4.1 High Level Group Structure
Figure 4.5.4.2 Final Chart

The Pinquu mobile application will use xml to retrieve information from the database for the following content: Login authentication; After login, user’s full first and last name retrieval; User’s picture and current status message - the “submit status” button will trigger the only event that sends information back to the database, and it sends the new message inputted into the textbox back to the database and updates itself accordingly; User’s list of projects; User’s project details including description, due date, collaborators, and post feed; Calendar events; and Document list.

The transition between how the different panels of the android application will interact is illustrated in figure below.
Figure 4.5.4.3 Transition diagram between the different android panels.
4.5.5 Hardware Modules and Status

4.5.5.1 Virtual Whiteboard IR Pens

Our IR Pen design is quite simple. Each IR pen’s internals consist of an ultra-bright IR L.E.D. and a push-button switch to activate it, and a portal battery pack to power it. We played around with multiple designs in finding the correct housing for the internals. Our very first design consisted of just tape and wire to hold the pieces together. This served as a very fast prototype for us to be able to hit the ground testing. However it is both not very visually appealing and doesn’t fit the traditional ergonomics of a regular whiteboard. Our second and third prototype attempt to fit the internal components in housings that more resemble a regular pen; one in a gutted expo marker and the other fitted in some over-sized straws. In the end both designs worked very well, but we went with the more cost effective housing of large straws in order to keep costs low.

The one variable part of the IR pens is the L.E.D. itself. We had to do testing with multiple brands and types of IR L.E.D.s in order to find one that had the best compatibility with the Wiimote’s IR camera. L.E.D.s all have different brightness, wavelength, and diffusion amount that make them different from other L.E.D.s. In our situation, we see the L.E.D.s with the highest brightness in order to make them more apparent in a wider range of lighting situations. For wavelength it seemed that 940nm IR L.E.D.s suffered from the least interference.

4.5.5.2 Status Stone

The status stone device, in its present phase of development, is made up of the following components: an Arduino Pro Mini microcontroller (ATMEGA168 processor), an ADXL335 tri-axis accelerometer, 2 Light Emitting Diodes, a Bluetooth Silver Mate module, USB Bluetooth host/adapter, and HOST (the host machine running the StatusStone software). The accelerometer collects z-axis data and is read via analog input by the Arduino Pro Mini microcontroller; A Kalman filter was implemented to remove noise from the sensor due to the low quality of the accelerometer. Through an internal proximity measurement, the microcontroller is able to detect when the accelerometer is lying right-side up (which maps to ‘AVAILABLE’) and when it is lying upside down (which maps to ‘NOT AVAILABLE’). Using this information, data is sent using the Bluetooth protocol wirelessly to HOST. On the device itself, when the microcontroller reads the analog data as ‘AVAILABLE’, current is driven to the green LED to indicate the status and the red LED is turned off. When the device is flipped and the accelerometer effectively inputs ‘NOT AVAILABLE’, the green LED is turned off and the red LED is turned on.

The StatusStone software stack running on the host machine first establishes communication with the external device – if the device is not discovered, the program alerts the user and then exits. Once the external device has been discovered, a ‘handshake’ protocol is initiated to ensure that the device is indeed the StatusStone and not some other device connected using Bluetooth. Once the handshake protocol passes, the program prompts the user for the kiva login information The program then prompts the user for their name, or in the general case, the person whose status the device is meant to be displaying. Once all this metadata has been collected, status information is ready to be generated – the internal software stack is initialized to ‘NOT AVAILABLE’ and the first server ping occurs when the device first reads ‘AVAILABLE’. It should be noted that some implementation details utilized to reduce glitches and ensure that false positives/negatives do not occur are not explicitly mentioned.
4.5.6 Software Modules and Status

4.5.6.1 Communication to Isolation Layer

The Virtual Whiteboard is an amazing tool, however it is most useful when integrated with the capability to save and upload the whiteboard files for later use. The whiteboard communicates with the Isolation Layer using HTTP requests, sending XML data which includes the Meeting which the Virtual Whiteboard files should be tied to as well as the file path of the Virtual Whiteboard so the Isolation Layer can properly grab the data and store it into an online database. The type of file to be stored into the online database is irrelevant; the storage system can handle any type of file. The files we commonly used in class were JPEG image files and annotated PDF files. Any number of files could be attached to a Meeting object using a Virtual Whiteboard object, so the database could store all revisions to the Virtual Whiteboard files leading up to the final version, allowing a user to see everything that happened during a Meeting.

The procedure to send the request to the Isolation Layer is quite simple, but requires a strong foundation for the backend. There exists one C# file, which we will call the HTTP packager, to specifically package an XML request into an HTTP request and send out the request, and returns an XML response from the Isolation Layer. It is up to the calling code to properly create the XML request to send out, and to extract information out of the XML response that comes back. The calling code comes in three sections. The first section is to create the outgoing XML request, which is then passed to the HTTP packager. The second section is to verify that the response from the HTTP packager does not contain an error. If that check detects no errors, then the third section is to extract data from the response. The response for saving a file to a Meeting is to have the ID of the Virtual Whiteboard object returned so the user can easily locate the file later, as long as he/she has the Virtual Whiteboard ID.

In order to properly support the Meeting and Virtual Whiteboard database objects, custom C# objects to mimic the database objects were created. When the program retrieves a Meeting or a Virtual Whiteboard from the database, it can temporarily store all the data inside a custom Meeting or Virtual Whiteboard C# object type until it is used.

The communication to the Isolation Layer can be further extended by introducing and implementing more XML requests that the Isolation Layer creates. The HTTP packager is complete and the backend foundation is clearly laid out: generate XML request, check XML response, and extract data from XML response.

4.5.6.2 Paint Program

The use of the Virtual Whiteboard can only replace the mouse in its current state. Thus, it can only mimic a whiteboard by using the paint program which is only operated by one user at a time. By exploiting the use of an altered version of Johnny Lee’s multipoint grid program, we are able to create a program in which multiple users may draw with LED pens simultaneously. Johnny Lee’s multipoint grid program allowed a user to use multiple IR pens to alter a grid. For example, using one IR pen and clicking on the grid, you are able to move the grid about. With two IR pens, you are able to perform various actions such as rotation and zooming in and out of the grid. An altered version of Johnny Lee’s multipoint grid was necessary because the altered version included a calibration feature which would allow us to be more precise when creating the drawing feature of the paint program.

The creation of the paint program involved discerning the method used to create the cursors. These cursors were used in the previous program to indicate position of each IR pen. The program was then rewritten to draw new cursors which now acted as individual points being
drawn by each poll of the Wiimote. These drawn points were then saved on the canvas allowing us to create a makeshift paint program that would continually draw points on the screen as long as the IR pen was pressed down. An eraser and a clear screen option was also added in as well as a save feature that would perform a print screen on the particular window and save it to the desktop. The eraser functionality basically would still be drawing as it were previously but in the same color as the background. The clear screen functionality was just a refresh of the canvas that we were drawing on. Multiple users are now able to draw simultaneously. The green lines indicated when only one IR signal was tracked and the blue lines indicated when a second IR signal was seen. Although multiple IR signals were tracked, as you can see with the first drawer’s area, there is no detection of which IR pen is being used as each IR pen sends the same signal and we can only determine which signal was seen first rather than assign each IR pen an ID.

Although the functionality was fine, there were a few limitations to this program. The drawing would possibly be a little choppy considering that it would just draw on each poll of the wiimote. Thus fast movement would not represent the functionality of a paint program. We were limited in the methods that we could create the drawing program as the initial base code required the use of vector drawing as opposed to using the Graphics class available for C#. We were also limited in the creation of buttons to use for options as there is no method of tracking individual pens when they are not on. When each IR pen is turned on, there is also only one state it can be in, which is on. Thus, there is no method of creating many different user friendly features for our program. This program could definitely be rewritten given a better understanding of the code base to perhaps increase the number of polls from the wiimote or completely revamp the canvas used to create a smoother and more user friendly program.

4.5.6.3 Status Stone Software

The availability of a user is seen by LED’s on the physical Status Stone. This availability is stored in the embedded software which is then stored in the remote server MySQL Database so it can be accessed by programs such as an IM Client.

The way this works is initially a user connects to the server that has the user’s information stored in the database through a socket connection that has been opened up. After connecting, every time a user updates his/her status physically via the stone, the stone sends a message to the Dispatcher (Isolation Layer) with the user’s current status. The Dispatcher is a program that acts as a middleman and will send/receive information to and from the database to anyone who requires it. The Dispatcher will then store the status information of the user in the database.

Specifically, there are 3 different messages the stone sends to the Dispatcher including connect, setStatus(available), and setStatus(unavailable). Each time the Status Stone needs to send a message it sends data over an HTTP port which was initially opened on the computer the Status Stone is linked to. Since the messages are specified by our protocol, it does not need to be dynamically generated (i.e. the user has predefined statuses). Therefore, 3 messages were created and are used throughout the communication. These messages are XML messages that are URL encoded and then put into a POST Request wrapper.

In the current state, there is only one way communication between the Status Stone and the Dispatcher because the Status Stone is the mastermind behind setting a status. In the future it would be possible to read messages sent by the Dispatcher regarding status changes, as would be the case if we let an IM client “compete” with the Status Stone to set the user’s status or to relay
any messages to a new version of the Status Stone (one that might have an LCD Display perhaps). The software could be improved as well to be more efficient as there were some problems with the socket connections.

4.5.6.4 Mobile GUI Application

The application was developed in eclipse using their android sdk plugin. There is one module called “pinquu.mobile” that has all the panels integrated together. Most of the look and feel is defined in xml, and the behaviors are coded in java.

At the end of the phase 3, I was only able to fully define the look and feel, fully define all function behaviors except for the filtering via pulldown menu on Spaces page, and did not have any information retrieval from the database. The reason for this is because I never got the dispatcher working on my development machine, and I needed that to access the SQL database information. At the demo, all the information was hardcoded.

There is one behavior in the android application that is not intuitive to the user. When the user navigates to the project page, they must go back to the profile page before they can navigate to the calendar or spaces page. This is the side effect of the way I implemented the project page. Instead of changing Context like how it’s done when user switches between the different tabs, I had the java code toggle between 2 different layers of display. That’s why when the application has the project page on visible and the profile page on invisible, it breaks how the tabs and their respective contents are correlated.

There is also still room for improvement in the interface. Currently, the different items in all the lists are supposed to take the user to the website or open the respective document, but because when the user touches the items there is no visual feedback, it’s not apparent that clicking on the items does anything. I should have implemented a feature such that the items will change on hover over or per click.

4.6 Isolation

4.6.1 Functionality

The initial design of Pinquu involved the individual software and hardware modules communicating independently with the database. While ideal during conceptualization, this initial design had a few intrinsic implementation flaws. First, it created a problem of code overlap between different teams. Due to the variety of a) code and b) possible means of communicating with the database, the code structure would have to be very specific. As a result, small changes in the database would necessitate changes in every related module. Second, it expected the database server to be able to handle all the requests smartly (be able to handle requests written in a variety of command formats). These flaws could have led to inefficiency and disorganization within the project.

In order to resolve these issues, Pinquu adopted Eric Rose’s proposal to include a proxy between the database and the front end, which would isolate the database and control query traffic. This isolating layer would take in XML commands from any user, perform the necessary database transactions, and respond in an appropriately formatted manner. Thus each team would not need database experts and the database had a layer of optimization protecting it.
4.6.2 Formal Specification

The Isolation Layer is a Java application consisting of 8 classes (groupings) and 61 packages (Java files containing a particular function). The following is a list of all the classes and their packages:

Group Class: Handles the project groups
Functionality:
- get all the members,
- adding a new member,
- deactivating an existing member,
- get all the posts from the group,
- get all the meetings of the group,
- get the info about the group,
- get all the files uploaded by the group,
- get all the links attached in the group.

User Class: Handles the users
Functionality:
- connect with the user,
- disconnect with the user,
- get the status of an user,
- post the status of an user,
- get the info about the user,
- get the groups that the user belongs to,
- get all the posts from the user,
- get all the meetings of the user,
- get the info about the user,
- get all the files uploaded by the user,
- get all the links attached by the user,
- get all the topics that the user is an expert in,
- get all the whiteboards that the user has written on.

Class Class: Handles all the classes
Functionality:
- get the groups that belongs to the class,
- get all the posts from the class,
- get all the meetings of the class,
- get the info about the class,
- get all the files uploaded by the class,
- get all the links attached by the class,
- get a search result from the class.

Friends Class: Handles the friends groups and lists (aborted)
Functionality:
- get a list of all the friends of a particular object (user, group etc.),
- get a list of all the friend requests made,
- accept a new friend request,
- ignore a new friend request,
- add a new friend,
- remove an existing friend,
- get a list of friends profiles,
- get a search result from the friends.

Meeting Class: Handles the meetings
Functionality:
- get the date and time of a meeting
- get the names of the creators and collaborators of the meeting
- get the whiteboard objects related to this meeting
- get the file-path to a particular meeting
- get a search result from the meetings.

Whiteboard Class: Handles the whiteboard objects
Functionality:
- get the file-path to a particular whiteboard object
- get the meeting that a particular object is connected to
- get the users that created a whiteboard object
- get a search result from the whiteboard

Web Class: This class handles the web objects (.swf files)
Functionality:
- create a new .swf file
- destroy an existing .swf file

General Class: If any particular grouping does not happen, the general class is used.
Functionality:
- get a particular post,
- get a particular meeting,
- get a particular file,
- get a particular link,
- get a whiteboard object,
- get a web object

File Class: This class creates and stores paths to ifolder files
Functionality:
- Delete a file path
- Store a file path
- Get information on a particular file

4.6.3 Experimental Measurements

At the moment, both the XML dictionary and most of the basic development of the code base is done. However, in order to confirm this both peer reviews and GUI tests are being conducted. Further, functionality is still the primary testing focus, rather than robustness.

Peer Reviews: Each member of the team was in charge of a specific set of program classes. Once the classes are coded they are reviewed (for syntax and usability) by another member. This helps resolve any obvious style errors not visible to the coder.

GUI testing: The Diagnostic GUI is then set up to test a specific module to check functionality. Further for each function, the XML input and the result is displayed. Comparing the GUI output to the expected result we are able to test the functionality.

4.6.4 Software Modules and Status

The Isolation Layer has a few functional specifications. Below is a brief description of how a regular command is processed.
- An incoming message is sent in precisely formatted XML as mentioned in the detailed specifications (in the appendix).
- The request is received by the Dispatcher, which parses the command and extracts the class and package referred to in the request.
- The request is then forwarded to the relevant subclass in a particular class. If the request is invalid, it is ignored.
- The subclass forwards the response to the class’s Command Handler.
- The Command Handler does the following:
  - Parses the command and finds the necessary parameters
  - Communicates with the database and processes the results
  - Forwards the data back to the package
  - The package formats the received data into the response XML format and sends the reply back to the front end.

As of the end of this phase, the following classes have been implemented, tested, and debugged.
  - analysis
  - class
  - file
  - general
  - group
  - meeting
  - search
  - user
  - web
  - whiteboard

4.7 Conclusions

4.7.1 Requirement Feature Table

4.7.1.1 HCI

None submitted.

4.7.1.2 Visualization/IOV

<table>
<thead>
<tr>
<th>Desired Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login Page</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Master Page</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Home Page</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Profile Page</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Project Page</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Search Result Page</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>BirdEye Result Page</td>
<td>Completed and Tested</td>
</tr>
</tbody>
</table>

4.7.1.3 Content Analysis

<table>
<thead>
<tr>
<th>Desired Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automated People/Topic Web</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Generation</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td></td>
</tr>
<tr>
<td>Automated Topic Generation</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Database Integration via Iso Layer.</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Alternate Default Arrangements</td>
<td>Not implemented</td>
</tr>
</tbody>
</table>

### 4.7.1.4 External Apps

#### Instant Messaging

<table>
<thead>
<tr>
<th>Desired Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chatting</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Group Chats/Conferences</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>File Transfer</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Availability Changing</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Authentication with Kiva Database</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Synching Friends with Database</td>
<td>Global friends list within company: completed and tested</td>
</tr>
<tr>
<td>Displaying additional info, e.g. projects and location, based on database</td>
<td>Friend list grouped by project: completed and tested</td>
</tr>
</tbody>
</table>

#### Document Sharing

<table>
<thead>
<tr>
<th>Desired Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Sync with local directory</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>History Tracking</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Group Sharing</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Admin Rights Management</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Authentication with Kiva Database</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Documents stored locally</td>
<td>Completed and Tested</td>
</tr>
<tr>
<td>Documents stored on external device</td>
<td>Documents are backed up on external hard drive.</td>
</tr>
<tr>
<td>Synchronization of file paths with Kiva stored file paths</td>
<td>Completed and Tested</td>
</tr>
</tbody>
</table>

### 4.7.1.5 Mobile Workers

#### Status Stone

<table>
<thead>
<tr>
<th>Desired Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluetooth Communication</td>
<td>Complete and Tested</td>
</tr>
<tr>
<td>Wireless Communication</td>
<td>Aborted</td>
</tr>
<tr>
<td>Physical Implementation</td>
<td>Complete and Tested</td>
</tr>
<tr>
<td>Status Toggling</td>
<td>Complete and Tested</td>
</tr>
</tbody>
</table>
Virtual Whiteboard

<table>
<thead>
<tr>
<th>Desired Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simultaneous Input from Multiple Sources</td>
<td>Complete and Tested</td>
</tr>
<tr>
<td>Calibration Mode</td>
<td>Complete and Tested</td>
</tr>
<tr>
<td>Clear</td>
<td>Complete and Tested</td>
</tr>
<tr>
<td>Save</td>
<td>Complete and Tested</td>
</tr>
<tr>
<td>Eraser</td>
<td>Complete and Tested</td>
</tr>
<tr>
<td>Integration with PDF Files</td>
<td>Aborted</td>
</tr>
<tr>
<td>Screen Sharing via SSH Tunneling</td>
<td>Complete and Tested</td>
</tr>
<tr>
<td>Physical IR Pens</td>
<td>Complete and Tested</td>
</tr>
<tr>
<td>Multiple Wiimote Integration for Accuracy Improvement</td>
<td>Aborted</td>
</tr>
</tbody>
</table>

Mobile GUI Application

<table>
<thead>
<tr>
<th>Desired Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Panel</td>
<td>Complete and Tested</td>
</tr>
<tr>
<td>Calendar Panel</td>
<td>Complete and Tested</td>
</tr>
<tr>
<td>Spaces Panel</td>
<td>Complete and Tested</td>
</tr>
<tr>
<td>User Authentication</td>
<td>Complete and Tested</td>
</tr>
<tr>
<td>Cross-platform Compatibility</td>
<td>Aborted</td>
</tr>
<tr>
<td>File Exchange</td>
<td>Complete and Tested</td>
</tr>
</tbody>
</table>

**4.7.1.6 Isolation Layer**

<table>
<thead>
<tr>
<th>Desired Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Class</td>
<td>Complete</td>
</tr>
<tr>
<td>Group Class</td>
<td>Complete</td>
</tr>
<tr>
<td>Class Class</td>
<td>Complete</td>
</tr>
<tr>
<td>Friends Class</td>
<td>Aborted</td>
</tr>
<tr>
<td>Meeting Class</td>
<td>Complete</td>
</tr>
<tr>
<td>Whiteboard Class</td>
<td>Complete</td>
</tr>
<tr>
<td>Web Class</td>
<td>Complete</td>
</tr>
<tr>
<td>General Class</td>
<td>Complete</td>
</tr>
<tr>
<td>File Class</td>
<td>Complete</td>
</tr>
</tbody>
</table>

**4.7.2 Summary of Key Design Issues**

**4.7.2.1 HCI**

The following are the key design issues we came across:

**Feedback and Availability**

One of the main features that we proposed in our visionary scenarios was to display the online availability of the different experts working in different projects. However, when we tested our wire frames with different users, many failed to understand how we were displaying the online status of a person. Using a colored rectangle is not good enough. Not only does it break existing chat conventions, but also there is no written feedback to indicate what the color indicates.
Terminology

Due to the lack of research that went into the project, some of the terminology used to describe certain aspects of the system was vague and misleading. As this project moves forward, we would recommend testing this system with an actual target audience to get a better understanding of what language should be used.

Affordances about the Wall of Projects

When we tested the “Wall of Projects” aspect of the system, we encountered some issues regarding its positioning within the system and purpose. The “Wall of projects” should be located in the global navigation instead of being part of a specific project page. The reason being that if it is located within a specific project page, it indicates that the information contained in this section is only specific to the project being viewed and not all the projects in the system. Furthermore, the “Wall of Projects” purpose needs to be refined. Currently, it only serves to view how other projects relate to the one you are currently working on, but measuring how a project relates to another needs to be determined. As of now it is very vague, and the users that tested our prototype did not see much value in it.

Navigation Breadcrumbs

Due to the complexity of the system, many of the users indicated in our user testing that they would get easily lost and no know how to navigate from one section to another. Therefore, we propose creating different ways to address this issue such as having navigation breadcrumbs. However, navigation breadcrumbs should be the only answer and other solutions should be explored.

4.7.2.2 Visualization/IOV

One of the first design issues we faced in designing our web application was determining the number of user controls to accomplish the functionality of each of the pages. This was important because this would provide us with reusable and efficient code, and also cause less overlap among work done by group members. An issue that I encountered with the home page was addition of functionalities that the Isolation Layer’s database did not have the capabilities for such as addition of posts and creating events. Thus, standard buttons were placed onto GUI without functionality. Finally, the largest and most important issue group faced was determining means of dynamically extracting data from the Isolation Layer’s database via XML Requests, and making sure the calls we were making aptly suited the code Isolation Layer had written for to pull info out of the past Kiva database. Since both Isolation Layer was working on this concurrently at the same time as our group was working on the pages, there were some brushes with delay in having readily available XML specs to complete functionalities.

When retrieving data from server, we must decide in what format we store data according to the way we parse data. For example, xmlDocument class was used for Gridview data while xmlElement class was used for basic profile information such as name, status, phone number, etc. The reason we have to choose among numbers of way to store locally is that xmlDocument class gives much more convenient way to store data as an object as traverse through nodes than xmlElement class does. For documents owned grid, single row is stored as one object with necessary attributes so that we can create an object for each row as we traverse xml file. On the other hand, basic profile information is stored in a class but we don’t need more than one. Thus, there is no need to make several classes for such data. In this case, xmlElement supports simple and effective way to parse data.
4.7.2.3 Content Analysis

The primary problem encountered by the Content Analysis group was determining exactly what HCI and others expected from our group. For the group, Phase 2 was primarily defined by unclear and conflicting objectives. While these problems were partially addressed, there still wasn’t a clear understanding of what was expected from Lingpipe, Birdeye and visual search capabilities. In Phase 2, most, if not all, of the Content Analysis groups’ abilities were questioned if they should be included in the project. With the decision to include these features, questions about how they would be integrated with the Pinquue webpage were important, but not clearly discussed and addressed. As a result, there was a continuing uncertainty about how our features would be integrated, used and executed by both users and administrators.

4.7.2.4 External Apps

Instant Messaging

The main issues we faced involved connecting the IM clients to the server as we were unsure how the domain and resource were specified. This was fixed after some trial and error: we discovered that the IM domain is determined by the name of the OpenFire server, and that the resource can be set by the user.

We discovered that Pidgin has its own cache, which we were unable to access. As such, we had to turn to Spark for Windows/Linux and Adium for Mac. After carrying out some user testing, we realized that an update to the groups on our Openfire server isn’t updated real time on Pidgin. It is, however, updated real time on Spark and Adium. We figured that we could do away with the additional plugins that Pidgin provides since the feature of having the groups updated real time was more important. Since both Spark and Adium have decent user interfaces and appearance settings, it was not a bad trade off.

At first, we had some difficulties linking our dummy local database with the Isolation Layer since most of our fields were named differently and the primary key indices of our tables were different from that of the Kiva database. For example, we used usernames as the primary key index while Kiva uses user IDs. This was solved by modifying an XML file in utilizing a custom database integration plugin for Openfire.

Document Sharing

The main issue faced in the last phase were specific software configurations that iFolder depended on in order to work. Also, we faced a problem with storing files on the larger hard drive. The motherboard of the server computer only supports one internal hard drive, which limits the size of the space per user.

In this phase, the main issues faced was determining how the files were stored locally on the server and finding a way to add users without manually clicking on a “Create User” button in the web interface. Both of these issues were successfully solved. Once we discovered that Simias encodes the folder structure, we knew that we had to go through iFolder’s web interface in order to determine each folder’s exact path. After much research, a command line tool was found which adds users without requiring the web interface.

4.7.2.5 Mobile Workers

The biggest issue we had during the last Integration and Implementation Phase was aborting much of what we were working on to make everything look good for the demo. Instead of development we shifted our focus on hard-coding the interface with desirable data so the demo would show the skeleton of a working version. Whether a hardcoded façade that looks
good or a well-developed but minimally functional project fits the definition of a prototype is debatable, but in the future, narrowing the scope of the visionary scenario to fit the length of the semester might help.

Another big issue stemmed from other groups switching their specs or interface layouts at the last minute. This affected our group significantly, as most of our work was physically implemented in hardware form, so changes were impossible or very difficult to make. From the beginning, hardware was not incorporated very well into the visionary scenario, so last minute changes were bound to happen. In the future, more emphasis should be given on even distribution of work over the semester, so these issues can be squashed early on in the process instead of the week before the demo.

4.7.2.6 Isolation Layer

4.7.3 Lessons Learned

4.7.3.1 HCI

- Communicating within interdisciplinary groups: It was a challenge to understand at first what every group was focusing on, and how it related to our specific group. Many times, the team would put some of the issues in hold because we were not sure if another group was resolving them. Also, there were some issues that fell between the cracks as one team assumed the other team was taking care of them.

- Visionary vs. Implementation: The HCI team was tasked in the first phase of project with creating visionary scenarios of our system. However, as we moved along the project and started researching what available technologies currently exist, we became more limited in our vision and had to cut back in many of the features that we originally proposed. Thus, the final demo was not as visionary as we intended it to be at first due to the technological constraints.

- Integration: The largest challenge that we encountered in this project was to integrate all the different aspects of the system. As we delivered the different wire frames to the other groups we had to not only explain to them what each wire frame was for, but also as a group we needed to figure out how to integrate it to the rest of the system, in order to have a cohesive system. There were many sessions, where an HCI member had to meet with other group members to explain the flow and logic behind the proposed design. This was not an easy challenge, as we had to make sure that our proposed design was implemented correctly or at least to the closest approximation.

4.7.3.2 Visualization/IOV

- When another group is assigned on backend side of project while my group is working front-end, communication is most important thing during the project. We assigned one person in our group to take care of xml specification for all pages, but it didn’t work well. We had many meetings with Isolation Layer group to ask for data and xml specification.

- Before jumping into coding phase, developers should have a big picture of project first. With big picture in everyone’s mind, next step is to design the data type, method. It is crucial because, most of web pages that we developed share a lot of data. Thus instead of coding same function or grid, developers should design on paper first.

- When coding with unfamiliar language, even with expert right next to one, one should spend a lot of time researching and practicing the new language. Most of our group had little
experience with asp.net and c#. Those who spent a lot time showed much more progress in every meeting than those who did not.

- Don’t use tortoise as some things don’t get updated. Tortoise could cause some problem when sharing files between Operating Systems among group members.

- Laying out design of user controls, external dependencies and database objects is very important before starting to make sure that all features are implementable.

- It is important to start as early as possible to get the best GUI possible. Otherwise implementation of web cannot be started as GUI design is not final.

4.7.3.3 Content Analysis

- It is crucial to have a clear understanding of what you are responsible for and how it is going to be integrated into the larger project. This is important enough that once future capabilities are researched, further work mostly should not be performed until there is a clear understand by all relevant parties of what the end result should look like.

- Any code relying on a separate teams code should be completed with a self-contained dummy data set so bugs can be found before the other group has completed and bug tested their code.

- People have many different preferences for displaying visual data; it is nearly impossible to satisfy them all. Here, “good enough” should be the goal as perfection is impossible.

4.7.3.4 External Apps

- Open source software may not be very well documented. The information provided is usually too basic to make use of. It takes many attempts at following various online guides before finally finding one that actually works.

- Server installation and configuration are much more complicated than expected since it requires a lot of in depth technical knowledge of databases and scripting languages. Perhaps the most important lesson would be that the latest version is not always the best as evidenced in our case since newer version might have more complicated bugs despite promises of more advanced features.

- Many things are much easier said than done. Our team has been running into problems with some of the more non-technical teams who do not understand the inherent difficulty in performing tasks that are easy to say out loud. This came up during this phase in the desire for the IM client to be embedded within the browser. Since we are working with open source solutions we are limited to what is available and it is nearly impossible to embed an already existing external IM client within the browser without completely starting from scratch. The problem is that it is so easy to say “just put the IM client in the browser.” It really requires a certain level of technical knowledge to make specification decisions.

- Open source software has the pros of being free and modifiable, but the cons of being badly documented, especially when it comes to dependencies and versions. This is especially true when it comes to specialized software: it has a smaller set of users and so it is much harder to find solutions to problems. Also, backwards compatibilities and dependencies involving different versions of different libraries gets extremely complicated to untangle. Sometimes, the only way to solve these problems is via (a lot of) trial and error, and to record your results online to help anyone who might be facing similar issues.

- Sometimes questions posted on the Kiva that were directed towards a group were left unanswered. I got better results when I targeted a specific person from the group and spoke with them face to face.
- Linux command options vary across different operating systems. I used a specific set of options for the “ls” command when testing on my Mac, but the option was not available in openSUSE and I had to rewrite my code.
- The Concept of communication structure of Dispatcher.
- Bash script.
- Open source software tools
- A miracle process in between implementation and integration
- Google is a very useful tool, as are message boards
- When developing a product, one should first focus on basic functionalities. Bells and whistles and prettiness should be of lower priority
- When time is an issue, people get creative and find workarounds for everything
- A lot of things seem simple in theory, but actually turning it into reality can be incredibly complicated
- It does not help to think of an over-complicated solution just to accommodate for unnecessary features for the user interface. It is more important to get the basic features and functionalities working first.

4.7.3.5 Mobile Workers

- Often times there are steep learning curves when starting a project from scratch.
- Know who you can contact for development when dealing with new software.
- Start early on integration because no matter how simple the components are, there is a high chance of failure as small problems percolate into large ones.
- For prototyping, we are pretty restricted when using the code base of interesting programs. Many features could not be edited the way we would have liked because of the method the original coder determined to use to perform their features.
- Clearly identify how all the database objects interact with each other before coding. I had completed the HTTP packager, but could not identify a clear use case scenario because the Mobile Workers vision of the Virtual Whiteboard objects differed from what the Isolation Layer imagined. This lead to a stalemate in opinions and by the time we finalized on a design and the module was implemented, it was too late to integrate into the final demo’ed project.
- Research into team members backgrounds first before assigning tasks (Some task assignments I made in the IOV group were mismatched with the team member’s background)
- Work more efficiently with the Isolation Layer. We didn’t properly identify which XML requests were high priority and thus certain requests were not completed when we needed them, requiring the IOV group to find workarounds which were less effective. It also produced a very stressful time for both groups to get things done as soon as possible with missing functionality, and that could have been avoided with better planning and communication.
- In a prototyping environment, often the milestones are set with priority given to demonstration of pseudo-functionality rather than complete functionality. It was difficult to admit to myself that I won’t be presenting a full package during the demonstration.
- Integration can be a pain, especially when it comes to cross-platform integration.
4.7.3.6 Isolation Layer

We did not have any centralized documentation for the code, so we ended up having to solve problems that had already been solved by other people. Having a developer wiki and adding information to it would have helped the development process greatly.
5. Project Management

5.1 Implementation and Integration Phase Results

5.1.1 Task Dependency Chart

5.1.1.1 HCI

HCI was not dependent on any internal group. We only depended on our users to provide feedback to improve the design.

5.1.1.2 Visualization/IOV

![Diagram of Visualization/IOV Task Dependency Chart]

*Figure 5.1.1.2.1: Visualization/IOV Task Dependency Chart*

Figure 5.1.1.2.1 shows the various dependencies for the Visualization/IOV team.
5.1.1.3 Content Analysis

The only task Content Analysis was dependent on was the Isolation Layer people and file XML request/response abilities.

5.1.1.4 External Applications

![Diagram of External Applications Task Dependency Chart]

Figure 5.1.1.4.1 External Applications Task Dependency Chart

5.1.1.5 Mobile Workers

![Diagram of Mobile Workers]

Kiva Database

ISO Layer

Multitouch Paint

Status Stone

Mobile Application
5.1.1.6 Isolation Layer
5.1.2 Summary of Work Log Hours for Phase 3

A - Admin, B - Architecture Design, C - Class, D - Coding, E - Data Migration & Restoration, F - Design Engineering, G - Field Testing, H - Group Meeting, I - Preparing Specs, J - Research, K - Staff/Leaders Meeting, L - User Testing, M - Writing Reports/Presentations, N - Website Design, O - Interface Design

<table>
<thead>
<tr>
<th>Team</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCI/GUI</td>
<td>3:30</td>
<td>0:00</td>
<td>63:40</td>
<td>0:00</td>
<td>0:00</td>
<td>0:00</td>
<td>2:00</td>
<td>37:15</td>
</tr>
<tr>
<td>Visual/IOV</td>
<td>5:30</td>
<td>0:00</td>
<td>72:05</td>
<td>210:20</td>
<td>0:00</td>
<td>0:00</td>
<td>0:00</td>
<td>18:30</td>
</tr>
<tr>
<td>Con. Analysis</td>
<td>0:00</td>
<td>0:00</td>
<td>10:35</td>
<td>17:25</td>
<td>0:00</td>
<td>0:00</td>
<td>0:00</td>
<td>3:30</td>
</tr>
<tr>
<td>Ext. Apps</td>
<td>2:30</td>
<td>0:00</td>
<td>42:10</td>
<td>46:00</td>
<td>11:10</td>
<td>0:00</td>
<td>2:00</td>
<td>7:00</td>
</tr>
<tr>
<td>Mob. Workers</td>
<td>1:00</td>
<td>0:00</td>
<td>84:30</td>
<td>149:00</td>
<td>0:00</td>
<td>3:00</td>
<td>12:30</td>
<td>26:15</td>
</tr>
<tr>
<td>ISO Layer</td>
<td>0:00</td>
<td>0:00</td>
<td>8:00</td>
<td>84:40</td>
<td>0:00</td>
<td>0:00</td>
<td>0:00</td>
<td>7:30</td>
</tr>
<tr>
<td>Class</td>
<td>12:30</td>
<td>0:00</td>
<td>281:00</td>
<td>507:25</td>
<td>11:10</td>
<td>3:00</td>
<td>16:30</td>
<td>100:00</td>
</tr>
<tr>
<td>Percentage</td>
<td>1.03%</td>
<td>0%</td>
<td>23.21%</td>
<td>41.90%</td>
<td>0.92%</td>
<td>0.25%</td>
<td>1.36%</td>
<td>8.26%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Team</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCI/GUI</td>
<td>0:00</td>
<td>6:00</td>
<td>0:00</td>
<td>23:00</td>
<td>53:15</td>
<td>12:00</td>
<td>2:00</td>
<td>202:40</td>
</tr>
<tr>
<td>Visual/IOV</td>
<td>7:30</td>
<td>4:00</td>
<td>2:30</td>
<td>0:00</td>
<td>37:05</td>
<td>0:00</td>
<td>0:00</td>
<td>357:30</td>
</tr>
<tr>
<td>Con. Analysis</td>
<td>0:30</td>
<td>0:00</td>
<td>0:40</td>
<td>0:00</td>
<td>6:45</td>
<td>0:00</td>
<td>0:00</td>
<td>39:25</td>
</tr>
<tr>
<td>Ext. Apps</td>
<td>1:30</td>
<td>21:20</td>
<td>3:00</td>
<td>8:00</td>
<td>50:45</td>
<td>0:00</td>
<td>0:00</td>
<td>195:25</td>
</tr>
<tr>
<td>Mob. Workers</td>
<td>0:00</td>
<td>5:30</td>
<td>3:00</td>
<td>0:00</td>
<td>31:00</td>
<td>0:00</td>
<td>0:00</td>
<td>315:45</td>
</tr>
<tr>
<td>ISO Layer</td>
<td>0:00</td>
<td>0:00</td>
<td>0:00</td>
<td>0:00</td>
<td>0:00</td>
<td>0:00</td>
<td>0:00</td>
<td>100:10</td>
</tr>
<tr>
<td>Class</td>
<td>9:30</td>
<td>36:50</td>
<td>9:10</td>
<td>31:00</td>
<td>178:50</td>
<td>12:00</td>
<td>2:00</td>
<td>1210:5</td>
</tr>
<tr>
<td>Percentage</td>
<td>0.78%</td>
<td>3.04%</td>
<td>0.76%</td>
<td>2.56%</td>
<td>14.77%</td>
<td>0.99%</td>
<td>0.17%</td>
<td>5%</td>
</tr>
</tbody>
</table>

*Percentage: time spent by class on each task/total time spent by class

5.1.3 Summary by Group of Individual Student Contribution

5.1.3.1 HCI

Ray Luong
- User research, Think aloud studies, Cog tool analysis, heuristic analysis, wrote introduction to user research in report

Nicole Fernandez
- HCI Editor, user research, think alouds, heuristic analysis

David Randall
- HCI Presenter, Drove project plan including establishing user testing timelines, Ran the final in-class demo, Wrote and iterated the script for the final demo, HCI Group presenter, Helped the UI/Visualization and Birdseye team on translation of the visionary scenarios and design questions, Worked with Rohit and on the status stone technical design, Completed the Status Stone industrial design and testing, Coordinated work with the IM team on client selection and plug-in.
Marcus Perez
- Developed script for demo, designed the GUI of Pinquu, User testing of final GUI

Shoshana Holtzblatt
- CogTool, User Research, heuristic evaluation, presented at final presentation

Chung-Yi Chi
- HCI Leader, user testing

Vikram Chatterji
- User Testing, mobile liaison, created wireframes
  
  5.1.3.2 Visualization/IOV

Ajay Ghadiyaram
- Software Engineer
- Home Page Development
- News Feed/Projects GridView Developer
- Report:
  - Software Architecture (Section 4)
  - Summary of Key Design Issues (Section 4)

Jaejoon Lee
- Presenter
- XML Specification Resolver
- Report:
  - Component (Section 4)

Jihoon Lee
- Web Developer
- Implemented Profile Page
- Report:
  - Summary of Key Design Issues (Section 4)
  - Task Dependency Chart (Section 5)

Paul Caravelli
- CSS Style Specialist
- Web Developer
- Worked on CSS for Every Pages
- Report:
  - Software Modules and Design Implementation (Section 4)

Paul Cho
- Leader
- Web Developer
- Worked on Every Pages
- Implemented Search Result Page
- Report:
  - Future work (Section 4)

Rachita Chandra
- Bug Tracker
- Software Engineer
- Implemented Project Page
- Report:
  - System Tutorial and Usage Scenario (Section 3)

Rantao Chen
- Technology Designer
- Worked on Every Pages
- Designed C# equivalences of Database objects
- Customer Authentication, log in, log out for Pinquu
- Communication to Isolation Layer for Pinquu
- Report:
  - Experimental Measurement (Section 4)

Suong-Sun Hong
- Editor
- Web Developer
- Implemented Login Page and Master Page
- Report:
  - Purpose (Section 1)
  - Background (Section 1)
  - Conceptual Design (Section 2)
  - Functionality (Section 4)
  - Summary (Section 5)

5.1.3.3 Content Analysis

Jason Lei
- Originally looked into how to change visualization of graph from a circular orientation to something more suitable for search (Turned out this was out of scope and HCI was satisfied with visualization)
- Created the people data structure so that Steve and I could work pretty separately from each other. My class would hold information about what exactly and in what format I needed from the Iso Layer
- Worked on code and algorithm to take information about people from Kiva and generate a people expertise graph. This linked both expertise and groups, though for the demo it showed only expertise
- Tested and worked on adding various features to the web such as adding a sidebar with different options to change graph, and also adding pinquu logo on top.
- Helped out with Steve and Rohith in getting all our code working together on the Lab machine
- Created a README for future people working on this where they would need to look in the code to modify the node click event.
- Summary of Work Log hours for Phase 3 and for All Three Phases section of the P3 report. Also worked on individual parts assigned by Steve.

Steven Luminais
- Responsible for interfacing with the Isolation Layer group.
- Trouble shooting with Isolation Layer bugs.
- Parsing the XML response from the Isolation Layer and formatting it in the format requested by Rohith and Jason in order for them to create Topic and PeopleWebs.
- Editor-in-Chief. Responsible for compiling the final Phase 3 report, tracking down missing sections, working with other editors to develop formatting standards, reviewing
Content Analysis reports for spelling, grammar and content, and completing missing, orphaned sections.

Rohith Salim
- I worked on the entire automation process for the Content Analysis Group. I went on to create an efficient script to run the different components in a sequential order. This way, the user would not have to run multiple different projects (through eclipse) and move things manually – the shell script takes care of that.
- I also modified the filter to reduce the noise that is created by LingPipe.
- On a leadership note, I was once again the presenter for Phase 3.

5.1.3.4 External Applications

Po Shin Huang
- Leader
- Part of the user synchronization in between iFolder and Kiva Database
- Report:
  - Suggestions to Improve Class
  - Future Work
  - Software documentation (Document Sharing)

Ivan Lee
- Presenter
- Research/User Testing:
  - Researched the Openfire API to find a possibility of integrating the KIVA users automatically and efficiently.
  - Wrote the configuration script in XML format and database script in MySQL to enable a custom dual database format
  - Added the ability to access the Openfire database in MySQL format in the administrator console via the web browser
  - Added the functionality for quick importing and exporting user data in MySQL format.
  - Tested the synchronization of the Openfire database with the Kiva database to ensure their interoperability.
  - Rigorous testing of the IM clients with the server backend
- Presentation: Compiled and presented for phase 3 for the external applications group.
- Report:
  - Software Architecture (IM)
  - Software modules and Status (IM)
  - Suggestions to Improve Class
  - Future Work

Cassie Li
- Research on curl and ways to autologin to a website
- Filepath Bash Script
  - Authenticates and logs into iFolder admin website, Saves cookie
  - Retrieves and parses data from the iFolder site (list of all iFolders and each iFolder’s owner and local file path)
  - Submits xml request to set the iFolder path
  - Retrieves file name, file path, and last modified time for each file in each iFolder
  - Submits xml request to set the file path for each file
- UserId Bash Script: Maps username to user id
- Report:
  o Experimental Measures (Document Sharing)
  o Software Modules and Status (Document Sharing)
  o Summary of Key Design Issues (for Document Sharing)
- Readme for filepath bash script and userid bash script

**Skylar Roebuck**
- Dispatcher troubleshooting
- Integration Conceptualization
  o Script to maintain updated users
  o Pull information from dispatcher loop through all users
  o Store data into separate txt documents
  o Parse txt documents for relevant information
  o Generate userList
  o Remove tags and delineate users
- Report:
  o Conceptual Design (Document Sharing)
  o Usage Scenario (Document Sharing)
  o Functionality (Document Sharing)
  o Suggestions to Improve Class

**Yiling Tay**
- Field Testing/Research:
  o Tested out the group adding and updating functionality on Pidgin and Spark, and decided that we should switch to Spark instead of Pidgin.
  o Did more testing on Spark to figure out what can and can’t be changed based on the backend database, and reported this to the rest of the group to help figure out how to tie in the database
- Presentation: Made slides for the presentation: the block diagram of our architecture
- Report:
  o Requirement Feature Table (IM)
  o Ranked Issue List
  o Suggestions to Improve Class
  o Future work

**Yumin Wong**
- Editor
- User Testing/Research:
  o User Testing on Adium and Spark
  o Signed in using existing usernames in our Kiva database and managed contacts, updated status, set profile picture, noted changes that were updated real time due to changes made on the server side and/or Kiva database.
  o Analyzed our local database tables created by Openfire using MySQL commands.
- Presentation: Screenshots of usage of IM Clients: Spark and Adium
- Report:
  o Compiled all the parts from each group member
  o Section 1 (External Apps)
  o Summary of Integrated User Interactions (External Apps)
5.1.3.5 Mobile Workers

Thomas Tzou (Leader)  
- Virtual Whiteboard Paint Program

Chris Jo (Editor)  
- Virtual Whiteboard Paint Program buttons

Rohit Banerjee (Presenter)  
- Status stone hardware implementation

Dan Lin  
- Screen sharing via SSH tunneling, IR pen construction

Rika Nakahara  
- Android mobile application for Pinquu

Rantao Chen  
- Technical design for Pinquu, authentication for Pinquu, communication to Isolation Layer for Pinquu, communication to Isolation Layer for Virtual Whiteboard, ordering parts

Veeren Mandalia  
- Status Stone Communication with Isolation Layer, PDF Annotation Tools

5.1.3.6 Isolation Layer

Kendra Garwin  
- Technical lead of the course project
  - Held leader position for phase 2 and phase 3
  - Point person for technical issues with the Isolation Layer

Skanda Mohan  
- Presenter for Phase 1, 2 and 3
  - Responsible for developing the whiteboard class

David Wang  
- Editor for phase 2
  - Responsible for Friends Class
  - Responsible for partial testing of the groups class

Andrew Yi  
- Leader for phase 2
  - Editor for phase 3
  - Responsible for implementing and partial testing of the groups class

5.2 Summary of Entire Project

5.2.1 Ranked Issue List

5.2.1.1 HCI

No reported issues.
5.2.1.2 Visualization/IOV

a) Need of better search algorithm as current one takes really long.
b) Need of XML specification for project top contributor and project description.

5.2.1.3 Content Analysis

a) Communication again was one of the biggest issues. At first I had thought that fundamentally changing the visualization of the graph was going to be a major concern after the demo since it was brought up in the demo and by some HCI students. I hadn’t clearly communicated with the professors and the new leader of HCI team about this, until we realized it was out of scope.

b) Birdeye
   a. A lot of debugging issues for setting up, and also using different xml specs to create different Birdeye graphs
   b. One thing that I didn’t have time to work on was a more meaningful node click event, for example redirecting someone to a separate page when a user clicked on a node in the graph. I would have had to talk more with visualization team. Instead I settled for creating README for students who pick up Pinquu in the future.

c) Topic Modelling through LingPipe is an extremely memory intensive process and even goes to exceed the default Java Heap Space. Whatever server is running the entire process should have sufficient RAM (2 GB) to ensure that good performance is maintained

d) Topic Modelling through LingPipe is also an extremely time consuming process especially when the number of files for analysis increases. Our group recommends that this is run by an individual on a bi-weekly or monthly process to see if any new topics of interest are generated.

5.2.1.4 External Applications

Instant Messaging
- Server clustering might need to be tested if we require a large scale solution to prevent a single point of failure.
- Had to change our choice of IM client from Pidgin to Spark and Adium since Pidgin has its own cache that doesn’t update the groups in the contact list in real time.
- Modified XML file to pull information directly from the Kiva database – provides real time synching with our own database.
- Openfire 3.7.0 should not be used since it lacks certain configurations necessary for us to set up a database that links to Openfire.

Document Sharing
- The correct version of the C# framework, Mono 2.4., was difficult to find
- Highly fragmented documentation that is not accurate
- Motherboard only supports 1 internal hard drive
- Trying to implement 2 server-based external applications on 1 machine adds additional complication and can create a choke point
- LDAP is unable to be configured
- Server configuration is nebulous at best and requires a great amount of guess work
- Documentation makes it nearly impossible to find the location of stored files
5.2.1.5 Mobile Workers

- Rika was originally going to develop the android application for Droid, but Motorola phones required a proprietary driver to connect the phone up to the computer, and that never worked so she had to resort to the backup plan of developing for the G1. Currently the app only works on the G1.
- The pico projector is not bright enough to display a visible image unless the surroundings are pitch black. There is a way to boost the brightness level through manipulating the factory settings, but this reduces its battery life to less than half an hour.
- The multi touch paint program has trouble drawing smooth images. If there is a way to somehow increase the polling rate of the Wiimote’s IR sensor, it would help smooth out the drawings.
- The current configuration of the paint program does not have a place to save its file onto the Kiva database. Integration with iFolder would allow us to save images directly onto the database.

5.2.1.6 Isolation Layer

No reported issues.
5.2.2 Summary of Work Log Hours for All Three Phase For Entire Class

5.2.3 Reflections on Work Log Hours and Distribution

5.2.3.1 HCI

HCI spent most of its time in class, group meetings, and writing reports. In class, we mainly met and discussed tasks that needed to be completed in order to move on. In our group meetings, we met to come to a consensus about design issues and who would be responsible for what. Report writing was a big component this phases due to our user testing. Everyone who performed user testing had to create a report on their findings to formalize what they learned. Also, the HCI group was again responsible for providing the narrative for the final presentation and compiling it, which took a lot of time.

5.2.3.2 Visualization/IOV

IOV group had a lot of group meetings within a group as well as with other groups such as HCI and Isolation Layer Group. This was necessary because with HCI group, we had to make sure we have the right design to implement for Pinquu. At the same time, we had to communicate with Isolation Layer Group to communicate with Isolation Layer to get data from the database. Meetings within IOV group member were important as each page of the Pinquu is connected in a way that results of one page lead to another page. Also Pinquu is designed
efficiently with reusing function and XML specification which lead us to discuss more within group to have the correct request and retrieve specification.

Unlike in other phases, most of the time was spent on web implementation. As no one had experience in web design, everyone started from scratch. By owing specific part of the page, each group member spent large amount of time implementing the actual webpage of the Pinquu as well as writing XML request and retrieval of the data from the Isolation Layer.

### 5.2.3.3 Content Analysis

Some members in our group are personally not a very big fan of the work log system. However, they do understand the importance of it. The workload was distributed pretty evenly. Steve, Jason and I each had our own individual component to work on so we all felt as though we put our share of work into the project.

There were weeks when I spent a lot of time on this class and there were weeks when I did not spend as much time. However, in the end each of our group members finished all the work we were supposed to have done.

Work logs were definitely hard to keep track of. I think it didn’t reflect. Though I logged hours of coding, I think there should be some sort of way to differentiate coding from debugging. Because that is what most of what I ended up doing. Debugging things that came up or that went wrong. I tried to reflect that in my work logs as much as possible by being as detailed as possible. But when they’re summed up and calculated into the table format, you lose a lot of contextual information on what kind of coding you were doing.

I definitely lacked in work logs in the first 2 phases which brought my total numbers down. I definitely tried to step it up in phase 3, but I still found it extremely hard, since I would be working till late debugging something, and work logs were definitely not what I had in mind when I went to bed.

Ultimately though I think I learned that in the real world, not only is your work very result oriented, but it’s very important to log your work well and cohesively. If I would do one thing different in the class from day 1, it would be to have had better work logs so that I could have better shown the professors my contribution to the class.

### 5.2.3.4 External Applications

We spent most of our time in class (26.4%), checking and discussing our progress as a group with the rest of the class. Class time is the best time to catch up with other groups face-to-face and to clarify certain doubts that we have with regards to implementation and user interface features desired by the GUI team.

The next category in which we spent most of our time is in writing reports/presentations (18%). Compiling works from different people can be really tedious, especially when our team is further split into the IM team and the Document Sharing team. It is sometimes difficult finding an overlap or similarities to fit both teams into the given report or presentation template. In addition, since each phase requires both a report and a presentation, it is no surprise that we spend almost a fifth of our time on them.

Next, we spent about an equal amount of time, as a group, on meetings and research (13.3% and 13.8% respectively). It is important for us to meet up to discuss our progress with each other so that we each know where we currently stand in terms of the given timeline for each phase. Conducting proper research is also very useful, especially when we’re implementing something that we may not necessarily be experts in. For example, the installation of software written by other people was tedious because it was open source and the documentation found
online was very poorly written or outdated. As such, extensive research and cross-references with other sources had to be done.

The last significant component is coding (10.1%). Since we had to deal with user authentication as well as interact with the Isolation Layer, we had to write code to request for certain information from the database, as well as write scripts to simplify the retrieval of all users in the database. Coding helps to optimize and speed up our interaction with the Isolation Layer. It was also necessary for our external applications purposes in maintaining a link with the Kiva database and updating relevant information in real time.

5.2.3.5 Mobile Workers

Overall, the Mobile Worker group spent the most time coding, with writing reports and presentations coming in second. However, just looking at Phase three paints a different picture – almost half of the time spent on the class was devoted to coding. This was necessary because even though we were dealing with hardware implementations, there was still a lot of software to write to interface the device to the computer, and then communicate to the isolation layer. In the future, it might be a good idea to allocate more time to field testing, as the focus of this class is in creating a demoable prototype, not a fully functional product. Because of this misconception, a lot of time was devoted at the end into hard coding data instead of building the core of the product.

5.2.3.6 Isolation Layer

If you analyze the summary work log table, Isolation layer does not have very many hours compared to other teams. This is mostly because the majority of our members did not fill out the logs until the phase has come to an end – when the hours have already been tallied and posted on the report. The majority of Isolation Layer time has been put into coding and about 1/3 of the time for gathering requirements (XML specs from other groups). Most of the interactions with other groups were to resolve XML spec issues and general questions about the dispatcher protocol. Kendra was the technical lead when it came to speaking with other group members. Everyone else primarily spent time coding different parts of the Dispatcher to make it functional for users to use.

5.3 Suggestions for Improving Class

5.3.1.1 HCI

- Some things could have been made clearer like exactly what tasks each group is responsible for and making this public to the entire class.
- Would have been nice to have real users to base scenarios off of from the beginning of the semester.
- Might have been better to stick with the same team the whole time or only change once. At least for project lead.
- A group that has the ability to coordinate and manage the overall goals of the project needs to be created as the ‘project manager group’.
- ‘Project manager group’ can morph into a ‘quality assurance group’ in the latter phases of the project. They would be able to coordinate the different tasks that need to be accomplished as well as to coordinate the various ‘open issues’ the groups come across while developing a system of this size.
5.3.1.2 Visualization/IOV

- Distribute student according to works that groups have. We had quite evenly distributed numbers among the groups. However, especially for phase 3, more students were needed for developing. When visual scenario was done, it was about time to shift focus to developing than drawing pages on paper. HCI group did tremendous job making visual scenario.

- During phase 2, IOV group was still waiting for mockups, and could not get start coding. IOV group coded few lines but without complete designs, it was hard for IOV group to code wild. Instead of spending phase 2 in designing mockups, maybe finishing designs by mid of phase 2 and start coding could be a better idea.

- Class could do better for improvement is discerning students’ abilities at the onset, and help everyone understand the technical roles of each group. This would allow for better planning so that the students can better formulate how to divide tasks so that each student is technically suitable.

- Have more design oriented competitions such as delta design to give student chance to reflect on group dynamics and where groups can improve

- Have professor/TAs attend more meetings in the later half if possible to give feedback on how things are progressing and things can be done in a better way.

- Have a liaison leader to ensure all liaisons are attending

5.3.1.3 Content Analysis

- This is one of the most interesting classes I have taken at CMU because the entire class is working on one single project. The class is then broken up into smaller groups based on the expertise and interests of individuals – much like how it is done in the real world.

- That being said, I feel as though the project we were working on was not as interesting as some of the work done by previous batches of this class. If more interesting projects could have been chosen, that would have been a lot better. Even then, I learnt a lot in terms of the how a product development cycle behaves and so on.

- There should be a more defined project goal explained in the first week or so.

- There needs to be a more efficient way to compile the phase reports.

- Points should be deducted from grades for report mistakes that occur multiple times by an author; there doesn’t seem to be any accounting for report feedback if it’s ignored.

- I thought the class went well overall. I remember at the beginning being a bit confused, given the speed with which things came up. But I thought this was all part of the learning process. I realize now that in the beginning it was deliberately vague what the professors wanted as an end-product, because it’s more about what the students want to produce and what they conclude from their research is possible. I wasn’t a leader during phase 1, but perhaps that could have been more stressed by the professors, so that students understood there was more freedom in the end deliverable.

5.3.1.4 External Applications

- My only issue with the class was the ambiguity of the assignment. I realize that part of the class was identifying the specific needs but a general framing of the project would have been nice to start off with.

- At the beginning of the course, I felt that I couldn’t see ahead more than a week at a time. It would have been nice to know what to expect in future phases.
- Start to manage open issues in Phase 2. This would help everyone understand what are other groups doing and boost the implementation speed.
- Should decrease the time period for phase 1 to allow for more time to work on phase 2 and phase 3
- More even distribution of manpower within the groups
- More feedback and interaction across groups that are interdependent.
- Provide a more defined assignment for the entire class to achieve.

5.3.1.5 Mobile Workers

- Pick a project next year that has a more hardware focus.
- Start the integration process early among different groups instead of waiting until the last phase to do so. There should be milestones along the way in terms of integration.
- Put a heavy emphasis on inter-group communication, because often the specs to a project were unclear and redundant work was performed.
- Hold people accountable for coming to meetings. The amount of delay caused by one person missing a crucial meeting has at times made the progress of things very slow, especially when those meetings are between different groups to discuss specs.
- The Kiva generates too many emails. Without the threading function in Gmail, the number of email we get is simply intolerable. Even with the threading function, if replies are made to different topics, Gmail creates new threads for each topic, and quickly deluges the inbox. Alter the Kiva so when a reply is made to any topic, an email with a generic title is sent to relevant people so Gmail can classify all of them into one thread.
- Narrow the scope of the visionary scenarios so we aren’t coming up with a last minute hack-job, but a stable, scalable and well-structured product.
- More emphasis should be given on the class working together as a group – due to the fragmentation, inter-group tensions ran high at times, and I personally don’t even know two-thirds of my classmates’ names.

5.3.1.6 Isolation Layer

- Make sure that students are evenly divided into groups based on their expertise. Isolation layer was definitely an integral part of the project, however we did not have as much man power as we would have liked.
- Started of slow, during phase 1, we felt we didn’t quite understand what the goal was for the project.