IT 11 046
Examensarbete 15 hp
Juni 2011

Enhancing the interactivity in the classroom via Smartphone

Mikael Åsberg

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Enhancing the interactivity in the classroom via Smartphone

Mikael Åsberg

In order to enrich the interactivity in the classroom a digital survey system can be employed to enable teachers to enquire questions to the students and review the answers in a quick and effortless way. Today at Uppsala University a commercial survey system is in use that consists of; handheld devices used by students to submit an answer, Wi-Fi adapters that receive the answers from the devices and software that creates and presents survey sessions. This report is concerned about finding a alternative to that system and is aimed primarily to improve the economical factor. The approach is to make use of the increasing amount of Smartphone devices that flourish among students today. This resulted into a solution that holds the basic features of the original system. The design employs a Smartphone application which lets the students select a survey from a content manager and then fill it out by interacting with an online survey service. The outcome of a survey can, similarly to the original system, be presented in an integrated fashion within a presentation slide to reduce the need of switching windows when showing a presentation.

In conclusion there is a basic and free alternative to the current system assuming every student has a Smartphone available. One of the advantages beside the economical factor include being easier to manage for the teachers as there is no need to administer response devices to the students prior to a survey session.

Handledare: Arnold Pears
Ämnesgranskare: Arnold Pears
Examinator: Anders Jansson
IT 11 046
Tryckt av: Reprocentralen ITC

Contents

Problem description

Background

In order to enrich the interactivity during lectures a digital survey system can be employed to enable teachers to ask students questions and review the answers from all the students in a fast and easy way. Today at Uppsala University a commercial system is in use, which consists of handheld devices used by students to submit an answer, Wi-Fi adapters that receive the answers from the devices and software that creates and presents survey sessions. This thesis is concerned about finding an alternative to that system and is aimed to primarily improve the economical factor. The approach is to make use of the increasing amount of Smartphone devices that flourish among students today.

Problem specification

The goal of this thesis is to document the results of producing a survey system with the following specifications: software that provides functionality for creating, editing and taking surveys and for presenting the result of those surveys in charts. To take a survey the participants should interact using a Smartphone application on their Smartphones that will let them select a survey and submit their response. The results should be easily displayed in a chart during the ongoing presentation held by the teacher. Further the system should be free and easy to use. The thesis will also include a comparison to the current system that is in use today.

Method

The project timeline starts off with a briefing of the existing system in use today to make a comparison to the new system possible. This is followed by an orientation phase, which is aimed at finding existing open source software that will ease the development of the desired system. The system can be broken up into the components shown in figure 1. The components are App, Survey, SurveyDB, Graph and Presentation. Survey provides functionality for managing surveys as well as collecting responses. SurveyDB is a content manager that contains a survey database that acts as a collector of all active surveys that teachers have activated at one particular time. It thus contains links to the Survey component. Graph holds functions to draw appropriate charts off the data collected in the Survey component. Presentation is a presentation tool such as PowerPoint that has the ability to show the most recent result of a linked survey, in order to reduce the need of having to change applications for that purpose during a live presentation. App is the Smartphone application that connects to the SurveyDB and directs the student to a survey. The component Survey sets the starting point of the orientation as this can be seen as the core of the system.

The sequent phase is proposal presentation, in which combinations of the results of the orientation phase is suggested as proposals for the system design. Each of the proposals includes a discussion that presents positive and negative sides of the proposal in question. One of these is finally selected accompanied by a comparison to the system specifications.

The last phase is the implementation phase where the selected proposal is implemented. A guiding through that implementation is described.

The result of each phase is presented in a chronological fashion in the next section.

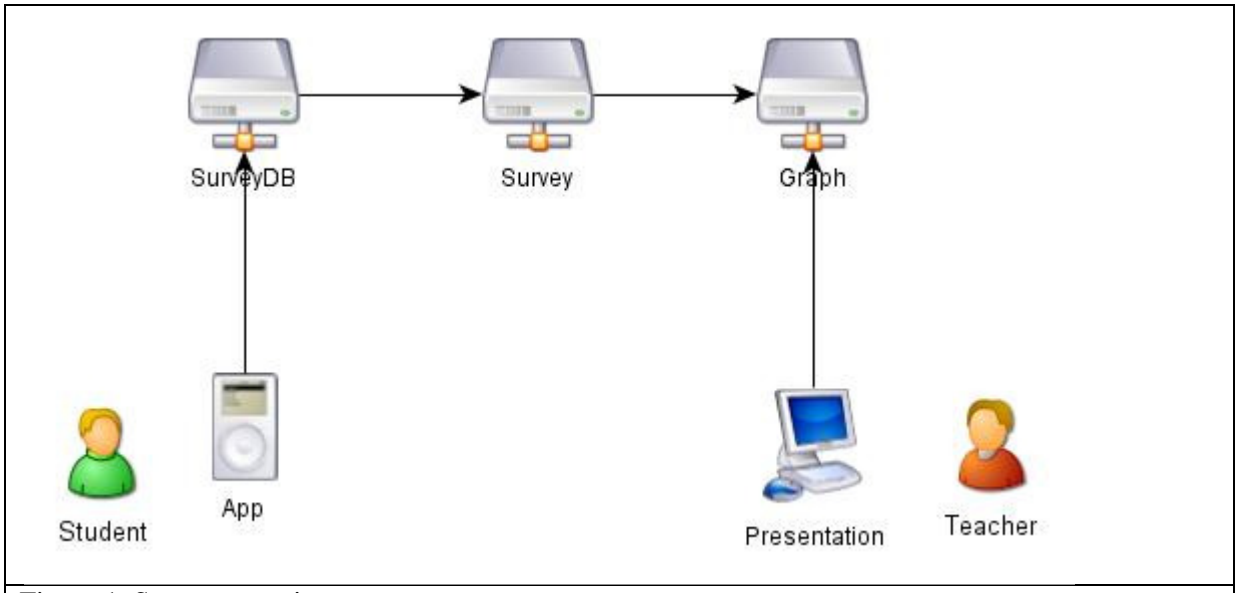

Figure 1. System overview.

Result

Current System

The system currently in use is from the manufacturer TurningTechnologies that specializes in audience response systems. They distinguish three basic elements in those systems: polling software, response keypads, and a response receiver. For each of those elements there are several options provided. The polling software lets the user author and present question and responses. One of those software products is called TurningPoint that enables integration with PowerPoint, by providing tools for displaying the questions and results within a PowerPoint presentation slide. The results can be shown as different types of charts and are updated dynamically in "real-time". The participants can be monitored by inputting the number of the unique response device that a particular participant is using along with additional identification information. This can then be used in the many types of reports available, which can be exported to Excel. TurningTechnologies also maintains a software product called TurningPoint Anywhere that is aimed to function outside PowerPoint by providing a floating toolbar, which can be placed in front of the window of the application in use. The response keypads are the devices used to wirelessly input a response to the software via the response receiver. The keypad in use is the model ResponseCard RF, with a battery life time of six to twelve months which may be considered when discussing the maintenance side of the system. [1]

Orientation

Here follows the result of the orientation phase where open-source software for the different components were browsed, investigated and discussed in relation to its potential applicability to the desired system.

Survey Component

The specifications on the survey component are mainly to provide different kinds of question types with a minimum of multi-choice and text input. Multi-choice is to prefer when the result should be easy reviewable in a chart, and text input when the students should be tested in a more qualified way that makes guessing harder. Further the survey component should be free

to use, as one of the main reasons of this system is to find a free option. Orientating in survey components resulted in the options presented below.

Google Forms

Google Docs offers Google Forms which provides a free survey tool with question types of multi choice, text input and ratings. Answers are exportable in Excel format. The results of the answers are retrieved in a table structure in Google spreadsheet and it is possible to view the statistics presented in charts. Further one can create own charts of different types that can be targeting a certain question, by making it point to a cell that calculates the sum of a specific string in the column where the answers are lined up. This chart can then be exported as an html image script or a JavaScript that will retrieve the latest update from the survey. [2]

One of the upsides of using Google Forms is that many users are familiar with Google Docs already, which is convenient out of a usability perspective as you will work more efficient when you recognize an interface you know of than having to explore something new. Further there is no need to maintain an explicit database for keeping the results of the surveys.

Now to a study on how Google Forms would integrate with the other components.
The SurveyDB would need a way of collecting the links to the Google Forms created by the teachers. One solution to this is to require the teacher to manually copy and paste the link into the content manager for the surveys. The manager would list all surveys sorted by teachers and their courses.

The Presentation component would need a link to either the statistics page of the survey or to another dedicated site that contains the exported chart. The straight-forward solution is to let the teacher copy paste the desired link to the presentation tool. The other solution is to have a fixed page that always display the statistics of the most recently activated survey. In that way there is no need to change the link in the presentation tool, provided of course that this tool holds this functionality. But this would require functionality to keep track of the most recent surveys and build charts on the results that can then are exported to this fixed page.

LimeSurvey

LimeSurvey is an open source project for surveys. It holds the ability to show statistics by showing charts of all questions, or just a subsection of the questions or the answers to specific questions. Also possible to view individual responses and export replies to Excel documents. The type of questions are many but included are all provided by Google Forms which is multi-choice, text input and ratings. Also possible to format the question in a variety of ways as it lets you edit using html code in the interface for creating questions. So one could for example include pictures associated with a question. By using a code functionality called tokens it provides an ability to make sure only certain people answers the survey. It offers functionality to send invitations to a group, and keep track of who has answered and make sure that they only answer once. The feature of keeping track of who has answered might not be needed in this situation as the requirement of all students answering might not be that strict. However the problem of making sure each participant is only answering once is desirable and is solved by a feature that enables cookies. This was something that Google Forms lacked, and might be useful in case some student deliberately votes more than once. But the ability to make sure only the student in class are the ones to answer the survey might be needed, and is easily solved by making one token that the teacher will tell the students when time is due for answering the survey. [3]

Now follows a study on LimeSurvey's demands on the other components of the system. The SurveyDB can in this case be merged with the Survey component, because all active surveys are listed publically at the domain http://mydomain.com/limesurvey/, where mydomain is the site where the LimeSurvey scripts are located. So the app component would then visit this link thus showing the user the surveys available at the moment. But as all teacher's active surveys will be located at this page it might be troublesome finding the correct surveys, so then one approach can be to adhere to a common naming representation that includes teacher name, course name and lecture number for example. Another approach is to make use of the fact that is it open source and sort the list on users. LimeSurvey has this point to its advantage compared to Google forms in which there was a need to copy the link of each individual forms to the SurveyDB.

The presentation tool needs a link to the statistics page of the survey, but this page is based on the input given in a prior page that lets the user select what statistics to show, and this selection is not included in the URL of the statistics page. Thus it is required to input information each time the result should be viewed and also it may require logging in, both of which would not be the case in the Google Form approach. As this is open source, however the code can be modified to bypass this. If one doesn't want to copy the link to the presentation tool it is possible to make LimeSurvey publish the statistics on a fixed page that the presentation always links to.

LimeSurvey requires a separate database that it can save the results of the surveys at. The database types supported are MySQL, MSSQL and Postgres SQL. Thus one obstacle with running LimeSurvey is that of installing and maintaining a supported database. In the Google case this was not an issue as it is handled by Google, which also has a better fault tolerance as it probably runs at replication servers.

Moodle

Moodle is a free course management system that educators can use to create and maintain online learning sites and is a well-used platform with millions of users. It requires a MySQL database to function and contains support for chat, forums, wiki, polls and databases, where students and teachers can upload any content for sharing of a large quantity of files. The poll function is relevant but it contains only single multi-choice questions, which is quite limited. The poll supports review of the results in a bar chart. Further it has a lesson module which is basically a series of HTML pages that contain a choice and a content area. The content area provides information and can end with a question. The choice area lets the student make some sort of choice that is required in order to continue the lesson. Depending on the answer the student might be directed to a different page than simply the next one in the series. For example if the wrong answer was picked, one could direct the student to the page containing more information about the topic of the question. The teacher has the ability to view each student's responses in detail once they have submitted, though the view doesn't support charts. Now this can be useful in cases where the survey/questionnaire needs to offer instant feedback to the student in case he or she answered a question wrong. As students perform the survey at different paces the reading of this feedback can be a useful way to spend the time that would otherwise be spent waiting for the students to synchronize. [4]

If Moodle is to be used in this system it would be a good idea to implement it as a course management system for the entire course, as it is entailed for this purpose, and saves the need of mixing with several systems when taking the course. However one could use Moodle solely for the survey purpose as well and then one Moodle site could act as the SurveyDB with links to each teacher's Moodle sites. The app would then connect to this main Moodle site at start-up.

PHPesp

PHPesp is a set of PHP scripts that lets you create and administer surveys, gather responses and view statistics. The software contains many types of questions including multi choice, text input, check boxes and drop down menus. It is managed online after a database initialization and just like Moodle it requires MySQL to function. The interface is not that exciting but that is because it is meant to be easily integrated with the user's own interface design. This software can thus be an interesting candidate to the system as the software is aimed for easy modification by providing a design that is modular and flexible. Unlike LimeSurvey PHPesp doesn't contain a predefined way of publically listing active surveys, but it can be modified for that purpose so that no separate content manager is needed to add and edit survey links. [5]

Survey™

This is a free web based survey platform that is aimed for Microsoft's .NET tools. It is written in asp.net and C#, which brings a little balance to the tools that have been presented which mainly have been written in PHP. It supports the usual question types including multiple choices, ratings, and plain text answers. The answers can be implemented in a point system that can be set together with conditional branching rules, similar to the lecture module in Moodle. The answers are shown in a report with pie and bar charts. SurveyTM is said to be easy to use that comes with a streamlined interface that is customizable. Just like LimeSurvey it contains support for disabling double submissions and to restrict a survey entry on an invitation code. The main drawback with this solution is that it requires a Microsoft server along with a Microsoft SQL server to store the responses in, which both are commercial products that oppose the system specification that the system should be free. However if the school already employs a license for these products that are used on other fields, this last note can be disregarded. Equally to PHPesp it doesn't contain a predefined way to list active surveys in public, so a separate page would have to be written for that purpose in order to not have to keep a separate content manager for the survey links. [6]

Presentation Component

This component is meant to support an easy way of displaying the results of the surveys whilst running a presentation, such as PowerPoint. The idea of having to switch to a new presentation tool just for this desired function may be a step some teachers are not willing to take. Therefore it is important to note that one can usually simply paste the link to the result directly in the presentation slide, and click on it in during the presentation which should open up an external browser automatically. However it is the switching between windows which is the inconvenient part which is what this component is trying to overcome.

Google presentation

Google has an online presentation tool called Google Presentation and as it also maintains survey and chart tools it would be convenient if this presentation tool allowed for embedding web pages. But that's not the case. It is possible however to add hyperlinks in the slides that opens a new browser tab when invoked by clicking. It is also possible to embed an image that are linked from an external source, though this image is saved at insertion and doesn't update when the source updates. Google supports import and export of PowerPoint files, which provides a smooth transition if power point is used for the teacher's existing slides. [7]

SlideRocket

SlideRocket is a free online based presentation tool that is closely integrated with Google, by for example offering the possibility to sign in using a Google account. It allows for creating and storing presentations online, with support for importing PowerPoint presentations. Further it supports transition animations similar to PowerPoint, embedding YouTube videos, chart tool and the option to choose between various graphic components to enrich the presentations. The most interesting part is however the chart tool that allows import of data directly from a Google Spreadsheet that can be connected to the results of a Google Form. The chart tool extracts data from the spreadsheet by using of one of the published formats that are available for a Google Spreadsheet; CSV which is the Excel format. The problem is that the published spreadsheet in the CSV format is not updated instantly but rather within a time frame of around five minutes (own measurement), so one has to be prepared that the chart won't be up to date at loading time. However this time period is a suitable time frame for the survey taking to take place, so in a potential system where Google and SlideRocket is employed the teacher could allocate around five minutes for such events. The chart types include among several types, bar and pie charts which are convenient when presenting the result from the answers on a multi-choice question. SlideRocket is free to use with the ability to upgrade in order to get access to further functions, such as sharing and collaboration and additional storage features. The premium status also offers the possibility to import Google Presentations. This would be quite convenient for teachers who utilizes Google's presentation tool for their lectures, as it allows for a smooth transition between adding a slide that holds survey functionality in their existing slides. The same effect can be reached by exporting a Google presentation in PowerPoint format from Google and then import the PowerPoint to SlideRocket, however this involves more steps. [8]

ZohoShow

ZohoShow is a free online based presentation tool similar to SlideRocket that also integrates with Google at the point that one can use their Google account to sign in and import supported formats from the same account. The formats supported for import are PowerPoint and OpenOffice. These formats and HTML are also available for export. Some of the functionality that differs from SlideRocket is the ability to embed raw HTML code in the slides. This means that, for example a Google Chart can be published as a HTML image code snippet that can then be inserted into a slide. The problem found was that the code is not rerun during the presentation unless manually refreshing the browser by pressing the F5 key. So in order to retrieve the latest chart one has to perform this procedure at the slide showing the chart. This only works if the format of the presentation is HTML which it is by default. One major advantage over SlideRocket is the waiting time between a modification in the spreadsheet and

the corresponding response in the chart that could up to several minutes in SlideRocket and is almost instant in ZohoShow. The reason is as previously mentioned that the published spreadsheet used for the chart in SlideRocket updates at intervals of a couple of minutes, while the source used for the image retrieves updates from the spreadsheet immediately. [9]

Power Point

PowerPoint is a popular tool for creating presentations. There exists functionality for adding web browser controls in the slides that can visit a web page at load time. A button can be inserted below the browser and a macro can be activated upon activation. This macro is to be written in Visual Basic and can for example perform the action of navigating to a specific page in the added browser control. Such a refresh button is desirable if all the responses have not been received yet. There exists a free Add-in that allows the adding of browser and navigation to a specified site automatic in a power point presentation. [10]

There also exists an online based version of power point available via Windows Live SkyDrive which is Microsoft's answer to Google Docs, providing the ability to share, create and maintain files online. It provides special functionality for the Office programs Word, Excel, PowerPoint and OneNote, so that these files can be opened and edited within the browser. Thus one can distribute their power point slides onto any computer with an internet connection which provides a high availability factor. A disadvantage is that in order to run the embedded web browser and its related macro the power point file has to be run on the external software, which requires the software in question to be installed on the presentation computer. However this is not necessary if the other approach is taken with a pasted link to the statistics page, though when the link is activated the page will be opened in a new browser window. [11]

Graph Component

This component is usually a part of a survey tool that allows the user to display charts of the answers received, however if one wants more control on how the chart is visualized, one should go for the approach of collecting the data from the Survey tool and then use it to create a chart. Here are a few open source options presented from the chart creation area.

Google Chart Tools

Google employs two kinds of charts, Image Chart and Interactive Charts that have different requirements and features. The image chart is a static image with a quick load time. To create own image charts one can use the Google Chart API, which holds functions to create and maintain image charts. The data and requests are sent to a server in a formatted URL and are thus limited to the size of a URL. The interactive chart is based on the Google Visualization API and supports use interaction via JavaScript. Allows for more fancy charts but has a bigger load time for complex graphs. This API also provides functionality to get data from web sites that supports Google Visualization Query requests, which would be useful if one want to collect data from a Google Form for example. Thus one approach could consist of the graph component being a separate server that fetches data from a Google form and displays the result in a chart using one of the described APIs. But Google Form and Google Spreadsheet also allows for creating and publishing these charts onsite, so another approach could involve simply copying the published html code in the desired presentation tool. [12] [13]

XML/SWF charts

This free software creates flash charts on web pages using XML files to store the data and modify the charts. This XML file can be generated dynamically with any scripting language. There are many possibilities to change the visualization of the charts. Flash animations and flash graphics allows for powerful development of the visualization. Supports full screen mode ability, which may be suitable if the presentation tool can't provide view of websites, so that the website containing the chart has to be visited explicitly in the regular browser. No flash programming is required which is convenient. All chart creation is done in the XML file so the HTML file containing the web page thus stays the same, which is good if one wants isolation between chart code and web code. [14]

Flot

Flot is based on a java script library called jQuery that according to the developers is a "fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development." It contains a range of modification possibilities in terms of visualization of charts and includes pie and bar charts. The fact that is uses JavaScript is convenient if one wants to use Flot in conjunction with the query functionality in Google Visualization API, which also supports JavaScript, as then there is then no need of mixing script languages. [15] [16]

SurveyDB component

This component is concerned with providing links to the active surveys in the system. Depending on the nature of the Survey component this component might or might not be needed to implement. Because if the Survey component already contains a built in function to list all surveys the Smartphone application can connect to the that page directly. The open source Survey components that don't provide this can be modified to publish such a list. However it could still be necessary to have several servers running the Survey component, in terms of fault transparency. So in case one survey server crashes, only a subset of the surveys are affected. This calls for a collector of the different server addresses. If one doesn't want to go for the approach where the Survey component list the surveys automatically, for example if the component in question doesn't support this functionality to be added, then the links for each individual survey requires to be added to a central server. In both of the previous cases means for manually adding links to the separate list page is needed. This requires login functionality which is preferably written in one of the two major server server-side script languages PHP and ASP as the popularity is usually a function of the amount of documentation available which will ease the development process. ASP requires windows server, but ASPChili and InstantASP can run ASP without windows. PHP however can run on most operating systems. As the system should be free PHP may therefore be preferable, as it can run on an OS that doesn't require a fee to use. However if the school already maintains a license for a certain OS this requirement can be disregarded. [17] [18]

In case the Survey component doesn't support listing and the teacher wants to reduce the amount of login tasks to the SurveyDB when he wants to add a new survey to the system, the survey link can instead be inserted into a separate page maintained only by the teacher. The SurveyDB component then simply associates the teacher with this remote location, and in this way the teacher only needs to login to the SurveyDB when there is a need to change this remote location.

Another option for the SurveyDB is using a free online wiki service such as Zoho Wiki. This site uses the wiki concept by offering an easy way to manage and share information contents

with others. It is possible to sign in using a Google account, which removes the need for having another account separately for the SurveyDB if the surveys are done in Google Forms. The downside is that there is no control of tailoring the interface so that it suits the application for the Smartphone. [19]

Another way is to design the whole component from scratch which offers a more tailored solution to the system, as one will have full control of the interface that can be fashioned to match certain requirements that the Smartphone app sets. The overall design is to maintain a database with information about each teacher such as name, surname, courses, links to surveys and meta data about those links, for example which course, lecture and date it belongs to. The index page is a PHP script that fetches and displays the teacher names from the database as links to a page with their courses listed. The courses are similarly listed as links to a page with the surveys of that course listed. At the page there should be a login button that directs to a login page where the teachers can login in order to edit their information.

The login functionality can be made smooth by using the Google Federated Login API, which lets users login using their Google account on third party web sites and applications. It is based on the open source project OpenID which frees users from having to set up accounts for different sites as well as the developers from having to maintain login authentication functionality. This is done by offering a framework to OpenID providers such as Google that makes the created accounts valid OpenIDs. These accounts can then be used by any sites accepting OpenIDs. The sites will then redirect the user to a login page at the provider. Upon signing in the user gets returned to the original site along with a consistent identifier that will be used to recognize the user. [20] [21]

App

The application component is the application that will run on a Smartphone, in this case an Android phone. Its' purpose is to provide a functionality to vote and take part of surveys in an easy way during lectures. Students will download the application once and then run it in class to take part in surveys announced by the teacher in class. One solution is to make it act as a browser that contacts a server that contains links to all active surveys. The student then selects one survey and the application directs the student there. The basic elements of a user interface on the android platform are called View classes. A solution that shows web content could make use of a View class called WebView which allows to maintain an own browser or simply to display some online content to the user. This puts demand on the interface size of the survey page as the Smartphone maintains a small screen. However the android platform has a standard procedure of calculating a target screen's pixel density and then depending on it being below or higher a certain constant, the screen is scaled in a definite way. [22]

Another solution is to interpret the structure of the survey that resides online, and visualize it in the app using GUI components in the android API. Such a component could be the so called widget, which is a View class that offers an interactive interface to the user, such as buttons, checkboxes and text-entry fields. Basically all that is needed to make a working survey interface. These existing widgets can be customized in order to create own action elements.

Solution Proposals

As the orientation phase is done combinations of the different findings are now presented. These combinations act as solution proposals to the system design.

Proposals on the System Design

Proposal 1

Here is an overall description of the first proposal: The app component contacts the SurveyDB which is a server that contains links to the surveys. These surveys come from Google Form, which is thus the Survey Component. The links to the surveys are added by the teacher to a dedicated page on the server running the SurveyDB component. This page is where the SurveyDB redirects users looking for surveys associated with the teacher. The Graph component consists of the statistics page in Google Forms, and the link to that page is copied to the presentation tool, which in this case is Google Presentation.

To add a link to a survey one would require login functionality so that a teacher can login and modify the relevant pages. The home page of the server then generates a list of the teachers' names and associates them with their pages.

Advantages
Minimum amount of mixed developers of the open source software could result in better opportunities for easy maintenance, as chances are higher that when the products are updated they will work in interaction with each other.
Disadvantages
A lot of manual work from the teacher; copy link to survey to the page and copy link to survey statistics to presentation. Also needs to maintain a separate server for the SurveyDB component. Google Presentation opens hyperlinks in a new window which might be a problem when running the presentation on full screen.

Proposals 2

The same as in proposals 1 except that when a Google form is created it is shared with a Google account dedicated for the SurveyDB component and owned by the administrator. The SurveyDB will then retrieve a list of its shared documents using the Google Document List API, and then find and display the appropriate surveys upon requests from the students. A solution is then to have one shared folder for each teacher. The Graph component is the page with the statistics of the survey in the Google form.

Advantages
Apart from those in proposals 1 only one copy task for the teacher; copy link from statistics page to the presentation tool.
Disadvantages
Need to maintain a separate server for the SurveyDB component as well as a Google account for the administrator.

Proposals 3

The same as in proposals 2 except that the SurveyDB will also retrieve the data of a Google form using the Google Visualization API and create a chart from the data. The Graph component can thus be any chart tool which allows for creating a wide range of different charts. It can then be published on a site that is dedicated for each teacher. So the presentation tool can always link to this site in the slide that is supposed to show the result of the survey.

Advantages
Apart from those in proposals 1, no copy task for the teacher. Free to customize the chart to one's likings by using any of the chart tools available.
Disadvantages
Need to maintain a separate server for the SurveyDB component and a Google account for the administrator. The update frequency of the graph is now dependent on the SurveyDB component instead of Google Form. So a valid question is how often one should update the chart.

Proposal 4

To overcome the need for a dedicated Google account for the administrator one could employ a script that is run by the teacher on the local machine when a Google Form has been created. This script is given the teacher's Google account details at installation and can thus access that account to retrieve the Google form, extract its data and draw a graph on a dedicated site. It can also find the key of the form and use it to create a link that it adds to the teacher's page at the SurveyDB.

Advantages
Apart from those in proposals 1, no copy task for the teacher, just the task of executing the script. Free to customize the chart to one's likings by using any of the chart tools available.
Disadvantages
Need to maintain a server for the SurveyDB and to keep the script updated with the latest Google account details.

Proposal 5

This proposal uses the open source LimeSurvey solution for the Survey component. The app in the Smartphone connects to the site that contains the LimeSurvey script directly, because LimeSurvey offers the ability to list all active surveys. So this saves the need for having a separate server for the SurveyDB. The teacher creates a survey by logging in to the Survey server, specifying the questions, setting its settings and finishes the creation by activating the survey. Then the statistics page for that survey is copied to the desired presentation tool. If one wants to display charts using other tools one can alter the code to make it interact with a tool of one's choice. The code can also be changed so it can publish the graph on some static page which the presentation tool always uses to display graphs of the latest survey. The list of the active surveys can be quite long which may make it hard to find the correct survey. But since it is open source this can be modified to be sorted according to teachers and courses.

Advantages
No need for a SurveyDB server to keep track of surveys. No copy pasting if modifications are done to the code so that the statistics always are shown on one fixed page. LimeSurvey contains more functionality around surveys than Google Form. Also as it is open source it can be tailored for best use.

Disadvantages
As there is only one server there is a single point of failure in the system, meaning that if this server breaks down the system is unavailable.

Proposal 6

This proposal also uses the LimeSurvey solution for the Survey component. But here the app in the Smartphone connects to a master server that contains links to several Survey servers that run the LimeSurvey script. These servers could be allocated on a one-per-teacher basis, so the master site associates a link to a teacher with that teacher's survey server. This gives one clear advantage in an increased fault tolerance, as there is no longer a single point of failure in the system. The problem then comes to setting the links in the master server to point to the correct survey server, which should be a dynamic process in order to make it easy to switch and add new survey servers. Thus it should implement a function for each teacher to login and set their links.

Advantages

Increased fault tolerance as a crash of a survey server just causes a subset of the system to be lost. If the master server went down however the Smartphone applications wouldn't be able to connect to the system. This would call for a replication process where another server can perform identical tasks when the main server shuts down.

Disadvantages

Need to maintain a master server to keep track of the survey servers. And if the system won't have a big user population this proposal may cause an unnecessary overhead.

Proposal 7

This proposal employs the online based service SlideRocket for the presentation component, which allows for creating presentation slides with embedded charts that are linked to a Google spreadsheet. So this is a solution that requires Google Forms to be used for the Survey component, as the results in a form are maintained in a Google spreadsheet. The format that SlideRocket requires from a spreadsheet is not directly compatible with the structure of the spreadsheet used in Forms, because the format defines series to be in the first row starting from the second column, and samples for each series to be in the corresponding column. However results from Forms are automatically put in the relevant cells and SlideRocket doesn't allow for choosing what cells to include. Therefore a new spreadsheet needs to be created that has the correct format and links to the results contained within the original sheet.

The advantage over the PowerPoint solution with the embedded browser is that SlideRocket, with the feature of being online based, has higher availability as it can be accessed anywhere on the Web, while the PowerPoint presentation has to be run with the PowerPoint software as it utilizes the macro functionality to maintain the embedded browsers. The PowerPoint Web App however is also online based but doesn't hold functionality to run the embedded browser. However it supports hyperlinks but they are opened in a new window when clicked, unlike SlideRocket which opens them in the same window.

The disadvantage is the waiting time between a submitted answer and the updated chart in SlideRocket, which is due to Google's automatic updates for published documents in certain formats being done in fixed intervals.

Proposal 8

The following proposal integrates the online presentation service called ZohoShow for showing presentations. It allows for adding hyper links and embedment of HTML code in the slides which can be used for example to display charts created in Google Spreadsheets or the

result page of a Google Form within an iframe. So by inserting the code for a chart that keeps track of the responses in a Google Form in a presentation slide the results of the survey will be available during the presentation. The slides do not update their HTML content during a presentation show however, so it is necessary to stick to the HTML format of the presentation which allows it to be run in the browser where the refresh button F5 can be pressed to update the content. An exception to the update point is slides containing an iframe which seem to update during the presentation. The advantage here over Proposal 8 is that there is almost no time between an income answer and an updated chart, considering one has updated the chart slide after an answer has been submitted. Further it is also possible to use other survey tools than Google forms by either including the link to the result page in an iframe or as a clickable hyper links in the slides. In the latter case the result page then opens in the same browser window as the presentation and if one wants to return to the presentation the backward key is pressed.

Proposal 9

This proposal addresses the solution in using the Smartphone app to display the survey user interface using widgets in the android platform, instead of showing the survey via a browser. A particular protocol is required as the app and the server containing the survey need to exchange data that the app can interpret in a correct way in order to display the appropriate text and widgets. The app then needs a predefined way to format the result of the input to be sent back to the server in order for the server to be able to present statistics of the participants of the survey. One way of achieving this is by integrating with existing open source survey software such as those described above. It could work in the following way.

The app contacts the survey server and retrieves a HTML page with the first page of the survey. Then the app searches the HTML page to find the input tools applied, and displays the corresponding Android widgets along with a confirm button. So there is a layer between the raw HTML page and the presented interface. When user presses confirm button the app finds out which input tools that received input and changes the corresponding input fields in the native HTML page, along with a JavaScript that simulates pressing the confirm button. After this is done the page is executed in a browser and the response would be identical to that of inserting input via the browser. This procedure is performed until the survey is completed. Upon completion the app uses the Web browser to display the real HTML page containing the list of active surveys. The main issue is to find the correct input elements and their Meta data in the HTML page. However by modifying the server-side software it can be sent in a nicely formatted way.

This solution would glue together with any of the proposals and open source Survey components described. Its advantages are an entailed user interface for the Smartphone, which would reduce the need of optimizing the page sizes of the surveys to make sure that scrolling over pages is less needed. This human-computer interaction perspective is important as the app should be easy to use for students coming from different backgrounds. The disadvantages are the extra layer of presentation that has to be included in the application, which adds maintainability work as updates made to the open source software might cause inflictions on the code.

Evaluation

A final system proposal is chosen by making a comparison between the system specifications and the previously described proposals.

The delay between response and updated graph should be as small as possible to be user friendly, and to allow for a more efficient use this result should be able to be embedded within the presentation in an easy way. Also the mobility perspective should be considered so that the presentation can be run and shared via the web browser. These requirements results in the use of ZohoShow as the presentation tool, because as it provides html code to be added in the slides it can embed an iframe that displays the statistics page of the survey.

The survey and graph component should be easy to use and being one of the market's biggest distributors of free software, Google has the advantage in that many of the potential users are likely to be familiar with it, and also is less likely to stop updating their software than a distributor with fewer users. Thus Google Forms is used for the survey component and its accompanied statistics module is picked for the graph component. This put the demand on the SurveyDB to be a collector of Google Form links, sorted by teachers and courses. The app component only has two proposals, and the simple one where the app just acts as a browser to a predefined link is chosen over an explicit presentation layer, because of time issues and the fact that it becomes sensitive to updates in the system and also is inflexible for change as it becomes very restricted to Google Forms as the Survey tool.

Implementation

At this stage a system design proposal has been selected and a guiding through the implementation of that design is here to follow.

Smartphone App

The app is primarily supposed to be browser based, meaning that it will contain a browser that will connect to the SurveyDB where a list of the available surveys are presented. The advantage over simply using the Smartphone default browser is that the user doesn't have to deal with typing in links. The browser widget supplied by the android platform is called WebView and is based on an engine called WebKit used for the Safari Browser. In order to use the WebView the android.webkit java package needs to be imported. The book Beginning Android from Apress will be used as reference in this section. [23]

The UI objects in Android are called activities, and this App would require only one activity that displays the browser. The layout is described within a XML file and the code used is described next. The first element is the root of the layout. It is here the other widgets will be contained. In this case, the linear layout was chosen because it provides a relatively straight forward layout solution that is suitable when the structure of the layout is not that complex. What it does is simply adding the elements after the other in either a horizontal or a vertical way. Further the variables layout_width and layout_height sets how much of the parent container the elements should fill. This activity only consumes a web view so it is set to fill all available space within the parent.

The manifest file is a XML file that describes what components and properties the application consists of. That is it describes the different activities and services the app provides. The first element in the manifest file is a manifest element that has holds the package attribute which should point to the base of the application. When a class in the application is referenced it is then not necessary to write out the base part of the class path. Below the manifest files it is possible to set user permissions, which indicates what permissions the app needs in order to function correctly. In this case as the app is going to access the Internet the following property needs to be added:

<uses-permission android:name="android.permission.INTERNET" />

The activities are added within an application element with properties for the class name of the activity and display name. Within this element an intent filter could be added to signal at what intents the activity should be displayed. The default intents that indicates the activity should be run upon launching is defined with the category element as shown in the code below.

```
<application android:icon="@drawable/cw">
              <activity android:name=".BrowserDemo2"
              android:label="BrowserDemo2">
                            <intent-filter>
                            <action android:name="android.intent.action.MAIN"/>
                            <category
                            android:name="android.intent.category.LAUNCHER"/>
                            </intent-filter>
              </activity>
</application>
```

Further an element sdk-min can be set to indicate what minimum version of the Android SDK the device that is to run the app should employ. This will cause the app only to install on compatible devices. Leaving the sdk-min out will set the version to 1. It is desirable to have a low minimum version as the app will then manage to cover more devices.

The source file of the activity is written in Java. It extends the Activity class to get access to methods and variables related to the Activity component. One of those methods is the onCreate method that is called when the activity is started. This is where the initialization starts and the user interface content of the activity is setup using setContentView(R.layout.main). The class R holds variables with paths to each file in the application. Layout.main refers to the XML file created for the main layout as previously described. A reference to the browser in the layout is then retrieved and its method loadURL loads a provided URL into the browser. It is important to include the http:// part of the URL otherwise the browser won't be able to find the page in sdk 2.3.

When a new URL is visited via the WebView it will by default ask the system to choose the proper handler for the URL, which could mean opening it in the main browser. As this is not what is desired a WebViewClient has to be created and assigned to the current WebView. This client class allows the host application to take over control when various events are happening to the WebView. After this client class has been instantiated and assigned to the WebView, a new URL visit will now be handled by the WebView itself as wanted. [24]

As the WebView doesn't come with any navigation tools by default it might be necessary in this case to provide a go home and go back functionality in case the wrong survey was picked, which would work out as inserting a button that is given an onClick listener that will be called whenever the button is clicked. This listener would then be implemented by overriding the onClick method which would verify that the clicked object is the button, before loading the new page in the browser. However since the site should be subject for good usability, there should be navigational tools embedded in the site as well. But having the same button in the site and in the app can raise confusion. So after a usability test the buttons in the app had to be taken out.

Presentation

The result of the survey can be embedded in a presentation slide using ZohoShow by inserting a HTML code snippet into a slide that either consists of an Iframe with the source set to the statistic page, or an image tag with the source set to a published chart from a Google spreadsheet. The latter involves a chart to be created and linked to cells that contain summarizations of the different answers. This requires more work so an approach to simplify this was taken by writing a Google App Script for the procedure, which is a JavaScript that provides easy ways to automate tasks between Google products and third party devices. [25]

The workings start with opening the spreadsheet connected to the Google Forms, which displays all the responses. By selecting an empty cell and inserting the code: "=COUNTIF(A:A;"answer"), the cell will show the number of occurrences of "answer" in column A. After doing this for each answer for a certain question, a chart can be connected to those cells and will then display the most recent responses for the question. The script took care of the coding part, by requesting the user for the column name that holds the answers for the question, and then ask for the answers that it should track in that column. When the script finishes, cells to the right of the question columns, contain the answer and their corresponding result in a nicely formatted way. The user then only has to select those cells and click the insert chart button to create a chart.

Survey Database

The Survey Database is a database of survey links connected to a web site that acts as a content manager for these surveys. The main page lists the available teachers as individual links to a list with the specific teacher's courses, which also acts as links to a list of the specific course's surveys. The surveys are represented by links to a remote location such as Google Forms. Login functionality is maintained for the teachers to be able to edit personal, course and survey information. Here follows a guiding through the implementation of this system that was built using the server script language PHP. Help and documentation about the language was found at the w3schools website, and is where the subsequent facts about PHP are taken from. [26]

As the system relies on stored information a database is the first component to consider. PHP is used as script language which has a convenient MySQL extension that makes database management easy to maintain, therefore MySQL is chosen as the database service. These tables were outlined after a database design process of the system:

Persons: name, surname, code, email, google-id, personID(pk), admin
Courses: courseID(pk), name, code, period, year
HasCourse: personID(pk), courseID(pk)
Surveys: surveyID(pk), link, courseID, date, lectureID

Table Persons has fields for name, surname, code; which is the code assigned each member by the university, email, google-id; which is used to identify the user when signing in with Google, personID; which is an internal auto generated ID that is used to identify a user throughout a session and finally admin which indicates if the user has admin rights.

Table Courses contains records for unique instances of a course. The fields are courseID; an auto generated ID, name; the course name, code; the course code, and finally what period and year this course instance is given. The relationship between Persons and Courses is many-to-many, so an additional table for this relationship is created called HasCourse. It has a many-to-one relationship with Persons, so a foreign key personID is included in the table. It also has a many-to-one relationship with Courses so a foreign key courseID is needed as well.

Table Surveys contains records of survey events. The fields are SurveyID; a link to the survey, a courseID, a lectureID and a date to identify what lecture the survey belongs to. Even though a survey can be used within more than one lecture (which can belong to any course), there is not much to gain by adding a table for that relationship with fields for courseID, surveyID, date, lectureID and Survey only containing surveyID, link. Because adding an existing survey would require the surveyID field to be filled with the ID of the requested survey, however as the only field that makes up a survey in the table Survey is a link, that foreign key field in the former table could instead be used for the link directly. So there is no storage benefit from it. Further two teachers that teach the same course instance will get listed the same surveys on the said course because there is no personID field in this table, and the search is done on the courseID. However this approach seems natural.

The next step in the design is to decide on what pages the system will consist of. As PHP is used to dynamically create the pages, only one page index.php is required. However because the login functionality is rather large it will reside in a class on an own page called login.php, which is also a consideration for the modularity perspective, in that it should be easy to change the login module without interfering with other parts of the system.

The index page lists all teachers as links with two get tags added to the URL as follows: www.mydomain.se/index.php?get=course&personID=teacherID, where get=course indicates that a course list should be displayed for the personID supplied in the next tag, where teacherID is the teacher's personID from the database. The index page will then check for that get tag using $_GET[“teacher”] to access it. If the get-tag called “get” equals “course” a query to the table Courses is made with the SQL statement:

```
SELECT * FROM HasCourses AS a, Courses AS b
WHERE a.personID = $_GET[“teacher”]
AND   a.courseID = b.courseID”
```

This will select all records in Courses that matches the teacher's ID. Each record in the result will be displayed as a link with the get tags “get=surveys” and “course=courseID” which indicates that surveys for the course with courseID should be fetched and listed. The name of the link will be a composite name consisting of the course name, code, period and year. So if the index page finds that the get-tag “get” equals “surveys” a query to the database is made on the supplied courseID resulting in the query statement:

```
SELECT * FROM Surveys WHERE courseID = $_GET[“course”]
```

The results are similarly displayed as links to the actual surveys, named by lecture id, survey name and date.

At the top of the index page there is a menu with the buttons: Back, Home and Login visible. Back goes to the previously visited page within the site. It makes use of one session variable

to store an array of each URL visited and another session variable to store the current index in that array to store the next URL at. After Back is clicked the index is decremented and the URL at that index in the array is visited. Home visits the list of teachers and Login starts the login process. When a survey link is clicked, the actual survey page is visited, and this menu then disappears, which can be frustrating if one wants to go back or log in, because then the user has to click the browser's back button instead, and additionally the site is going to be viewed via a browser without navigational tools in the app. This calls for including the menu when visiting the survey site as well. So to solve this problem the survey link now directs to the index page with a get tag set to the id of the survey. The index page will then find the associated link from the database and insert it into an iframe. An iframe is used to display a website within a website, so the menu bar will be visible in the main page while the survey is visible below it.

The login functionality makes use of a class called GoogleOpenID that resides in login.php which includes a piece of open source code under the GNU General Public License, that makes use of the OpenID functionality provided by Google, by directing to a login page for Google. After signing in to a Google account, the user is directed to a predefined return address, which in this case is index.php?stat=signed. The index page checks if this tag is set and if that is the case it retrieves the result of the response that Google returned from the login process. The result consists of a consequent ID and an email address used for the Google account. If the email exists in the table Persons the login process proceeds, otherwise it is cancelled. An identifier is stored in a session variable by the following PHP command: "$_SESSION['user'] = userID." A session allows for values to be saved between pages in the application, starting when a user enters the application and is deleted upon leaving, i.e. closing the browser. It is used to identify the user throughout the visit. Once signed in the menu changes to reflect the actions the user can undertake. If signed in for the first time a form to add personal information is displayed. [27]

More particularly the menu changes to the buttons: Back, Home, Edit user and logout. "Edit user" will display the current information along with an input form that was visible when the user signed on for the first time. Logout will make the current user logout by destroying the PHP session. Further when viewing the list of the user's courses it will be accompanied by an edit form and a radio button for each course. By selecting one of the radio buttons, inputting data in the fields and pressing a new button called Update, the selected course will update accordingly. If no radio button or the default one is selected when pressing update, a new course will get added. When the survey list is displayed for one of the courses held by the current user, the survey list will, similarly as for courses, be constructed with radio buttons and an edit form. If the user is an admin an additional button labelled edit accounts is displayed, which directs to a page that offers possibility to add and delete accounts.

The design issues that was regarded in this web site conforms partly to the requirement set by the Smartphone devices, through which the site is most commonly going to be used. Since the phones usually have a limited screen, it is therefore necessary to make the relevant part as easy accessible as possible, to avoid the need of scrolling. The relevant part for the students is the lists, and they are placed at the top left corner to minimise the risk of having to scroll horizontally to view them. The menu is made small, and placed above the list to be easily recognized and is easy available as it too doesn't require scrolling to find. The visual elements are much retracted as there are no images in order to reduce the load on the internet connections to the phones, which can often be limited in terms of bandwidth and cost. If the survey link leads to Google Forms a tailored approach is followed to adjust the layout for the

site. Google Forms employs an embedding function that uses a minimal layout design, with the purpose of not disturbing the existing design on the page it is embedded in. One uses this by changing the string “viewform” to “embeddform” in the URL to the Google Form. The latter makes the questions more left aligned than the former, and also lacks certain visual elements which is suitable for the cheap approach needed for the Smartphone. The URL to the “embeddform” can be copied manually by the user from Google Forms which has an existing menu labelled ‘embed form’ that displays a piece of HTML code consisting of an iframe with the source to the survey. However as the site only accepts links, merely the source of the iframe has to be copied. This approach is a bit tedious and can be troublesome for inexperienced users. One solution is to allow for both types of Google Forms links to be supported, so that the chance of achieving success on the first attempt is heightened. The “viewform” link is converted to embeddform by simply looking for “viewform” and replacing this substring of the URL as mentioned earlier. The “embeddform” link and other links are not modified. Then when clicking on a survey link, an iframe is created with the source set to the relevant link. A list of design principles from Designing Interactive Systems: People, Activities, Contexts, Technologies, David Benyon was considered during the design process. These principles guide a designer by orientating to key features and important issues in a design. One of those is “navigation” meaning that the user should be able to easily navigate in the structure of the system and thus need to be aware of the current location in the structure. This is followed by printing a relevant headline above the displayed lists, for example the name of the teacher when listing the courses that belong to that teacher. Also the back button provides for easily navigating to a previous page and a home button to get to the home page in one click, which also adheres to the Flexibility principle which is saying that there should be several ways of reaching a goal. Further the principle “feedback” was considered which means that the user should always get feedback on the performed actions, and this is conformed to by always printing a message on the outcome of an update operation and on the login status. Also the consistency principle was adhered to which refers to the fact the design should be consistent throughout the system. This was followed by keeping the format of the lists the same in all of the three list types: teacher, course and survey. Also the format of the edit forms for the same types is consistent. This principle was also considered when choosing the Google Login function. As the users are meant to create Google Forms which requires a Google Login, and then copy the link to this content manager which would require an additional login, it would be in line with good usability to include the same login interface for both sites in order to keep a consistent manner in the bigger system image as well. [28]

Guide

Here follows a guide for the teachers in how to use the system.

1. Setting up a survey

Go to www.google.com and sign in using your Google account. Follow the menu item Documents. In the menu “Create new” press “Form”. Now edit the question and answers that are going to be part of the survey. When done, you copy the link to the published form. In the edit view it is displayed in the lower part next to the description “You can watch the published form here”. It is also accessible from the spreadsheet view in the menu “open real-time form”.

2. Publish the survey

The students now need access to the URL to the created form. Either you can show the URL explicitly in classroom or you can use the content manager in the following way:

Go to the content manager for the surveys (ask your administrator for the address). Click login and you will be directed to a Google login page, where you login using your Google account. After having logged in you will get returned to the content manager and if the admin has added your email to the system you will now be signed in. If it is your first login you will be asked to fill in some personal information. The home page displays a list of teachers where your name is highlighted. Clicking on that will display a list of courses that you are teaching. You can add, modify or delete a course. When clicking on one of the added courses a list of its surveys is displayed. And to add a survey, you simply fill in the appropriate information such as name, lecture ID, and date. In the link field you paste the link that you copied at the end of step 1. When done you log out by clicking logout.

3. Retrieve the result in a presentation

If the result should be embedded within a presentation slide, ZohoShow is recommended. Otherwise the results can be displayed by browsing the result page of the created Google Form explicitly, which is done by going to the created Form and clicking on a menu item called Show Answers. This has two sub items: “conclusion”, displaying statistics of all the questions and “spreadsheet” that displays each answer in a Google spreadsheet.

- To embed the statistics page of all questions: First sign in to ZohoShow at http://show.zoho.com/login.do using your Google account and go to your slide. In the menu “insert” there is an item called “HTML Code”. Click that and paste the following code snippet into the pop up window:
  `<iframe src="SOURCE" width="100%" height="100%"></iframe>`
  where SOURCE is replaced with the URL of the statistics page for the Google Form. After pasting it into the pop up window and clicking OK, you may want to resize the window that appeared on the slide to your likings. An alternative approach is to click the menu “insert link” and paste the same link into the pop up window. Now during the presentation you have to click the link explicitly to get to the result page, and then go back using the backspace key. This last approach can also be done with Google Presentation, PowerPoint online and SlideRocket to mention a few.

- To embed the statistic of a single question: Go to the spreadsheet with the answers for the Google Form. There you find cells for each question in the first row and the answers in the columns below. Select any empty cell to the right of those columns, and insert the following code snippet:

```
=COUNTIF(A:A;"answer")
```

  Where A is the column letter of the column with the question and answer is one of the possible answers to that question. This formula will count the occurrences of those within the specific question column. Thus you will need one cell for each answer. The cell above each of those cells should contain the corresponding answer that the cell below is counting, in order to ease the chart creation. When done, you select all the created cells and click the insert chart button in the menu. Here you can select among a variety of diagrams, but the bar and pie graphs work well here. After inserting the chart, you can click on it, select “publish chart” and image format. Copy the code snippet that looks as:
  `<img src="URL" />`
  where URL is the specific link to your chart. Now sign in to ZohoShow at and go to your slide. In the menu “insert” there is an item called “HTML Code”. Click that and paste the copied code snippet into the pop up window.

Discussion

Here follows an investigation on how the problem specifications were met and how it compares to the current system.

"The user should be able to select a survey using their Smartphones." This is manifested by the content manager for the surveys and an application that connects to that manager. The content manager contains functionality to add and edit user, course and survey information, so that the surveys can be sorted according to teacher and course. In the current system there is no need for a content manager, as the questions are presented in class and the responses are made via the response keypads to the Wi-Fi response receiver. The need for a content manager can be made redundant in this system as well however by making a "general" survey that only includes numbered answers that thus can be tied to different questions which are then shown separately during the presentation. This means the teacher has to reset the survey and possible save it after each session, but it seemingly makes the content manager redundant if it only takes one survey link per teacher.

The app works by visiting the manager in an application-specific browser called a WebView. The same result could be achieved by typing in the URL to the manager in the Smartphone's main browser; however this approach eases the burden on the students and allows for a quicker setup. Future extensions could involve adding a presentation layer to the application that interprets the different types of questions, may it be multi-choice, text or rating, and displays the corresponding widgets from the SDK in use. This allows for a more tailored GUI that eliminates possible need for scrolling horizontally. In the current system the response keypads in use added cost of maintenance, organizational overhead in distribution and collection of the devices, as well as a restriction in the number of participants. By replacing those devices with the students' Smartphones several advantages are gained; the cost of maintenance is now put on the students, the organizational overhead and the need to educate the students about the devices are eliminated. Also as many of today now uses Smartphones it is arguably a less restriction on the number of participants that can take part in a survey.

The specification of editing and taking a survey part was met with the online service Google Form as it does meet the basic requirements by providing functionality to create, edit and take a survey. However certain extensions may be desirable, such as a function that will prevent users from submitting several answers in a row. This can be done in the content manager by creating a cookie when the user visits a survey. The cookie would include the id of the survey and have an expiration time set to an optional time that could be decided by the user who supplied the survey. A user will get denied when trying to visit a survey which has an id equal to the value of this cookie. The result of the survey should according to the specifications be viewable from a graph, and Google Form provides this via a statistics page over all the questions. If one only wants a result page of a single question, it involves adding a Google app script which is a bit tedious. However this need may not be that common. In the current system there are a lot more tools and functions available, such as limiting the participants, non-anonymous participants and a wide range of statistic tools and graph representation.

Further the specifications said that these results should be viewable within a presentation. One way of achieving this is to use any presentation tool that allows a hyperlink to be added to the slides, however then the link has to be explicitly clicked during the presentation and some tools such as Google Presentation will open the page in a new window. A way that doesn't

require these extra steps is the online based presentation service called ZohoShow that allows for including HTML code in a slide. So by inserting the URL to the results in an Iframe code snippet, the result will load upon viewing the presentation slide. The page doesn't update dynamically however so pressing the key F5 or revisiting the slide is required. This is not necessary in the current system that employs a full integration with PowerPoint, which updates the graph in "real-time". However one can argue if this is really a necessity. Showing the results to the participants in real time may affect their response as they take input to what others are voting.

About the usability aspect of the system, it is fairly straight forward to create a survey online in Google Docs, and as the content manager is focused on the concepts around surveys and lectures and merely provides basic functionality, it should be quite easy to get started. The usability for the Smartphone application is related to the content manager as this is where the app user is directed. One additional aspect however is that the embedded type of Google form is used before the regular view with graphical elements, in order to respect the Smartphone requirements of less data traffic and restricted screen size. In the current system there is a need to install software and maintain the response keypads and response receiver, which requires some additional steps that could be tedious for the non-technical user. An overlook of the toolbars and menus in the software indicates a good usability feed, with relevant icons that support the purpose of the particular button. One problem with so many features that this software provides is that it can be hard to find the relevant function and/or there is a bigger risk of doing something faulty. And if the main purpose of the survey is of basic nature, that is creating and publishing an open anonymous survey and viewing the results, perhaps this system provides too many extra features that will overwhelm the beginner, making the setup cause a lot of friction.

In conclusion there is a basic and free alternative to the current system assuming the participants have a Smartphone available. The survey software and server is maintained for free by Google and the cost of maintenance related to the handheld devices is transposed to the students via their Smartphones. So by making use of the increasing amount of free online based services an organization has the potential to create more cost-effective solutions.

References

[1] http://www.turningtechnologies.com/ 2011-05-13
[2] http://docs.google.com/support/bin/static.py?page=guide.cs&guide=27248 2011-03-30
[3] http://docs.limesurvey.org/ 2011-04-20
[4] http://moodle.org/ 2011-04-20
[5] http://freshmeat.net/projects/phpesp/ 2011-04-20
[6] http://survey.codeplex.com/ 2011-04-20
[7] http://docs.google.com/support/bin/static.py?page=guide.cs&guide=19431&topic=19433 2011-05-13
[8] http://www.sliderocket.com/ 2011-05-13
[9] http://show.zoho.com/ 2011-05-13
[10] http://skp.mvps.org/download.htm 2011-05-14
[11] http://explore.live.com/windows-live-skydrive 2011-05-13
[12] Google Chart Tools, http://code.google.com/apis/charttools/docs/choosing.html, 2011-04-20

[13] Google Chart Visualization API, http://code.google.com/apis/visualization/documentation/using_overview.html, 2011-04-20
[14] XML/SWF, http://www.maani.us/xml_charts/index.php?menu=Introduction, 2011-04-20
[15] jQuery, http://jquery.com/, 2011-04-20
[16] Flot, http://code.google.com/p/flot/, 2011-04-20
[17] Requirements PHP, http://www.w3schools.com/php/php_install.asp, 2011-04-20
[18] Requirements ASP, http://www.w3schools.com/asp/asp_intro.asp, 2011-04-20
[19] Zoho Wiki, http://www.zoho.com/ , 2011-04-20
[20] Authentication and Authorization for Google APIs, http://code.google.com/apis/accounts/docs/OpenID.html, 2011-04-20
[21] OpenID, http://openid.net/ , 2011-04-20
[22] Android WebView, http://developer.android.com/reference/android/webkit/WebView.html, 2011-04-20
[23] Beginning Android, Mark L. Murphy, Apress 2009
[24] http://developer.android.com/reference/android/webkit/WebViewClient.html, 2011-05-14
[25] http://code.google.com/googleapps/appsscript/ 2011-05-14
[26] http://www.w3schools.com/php/default.asp, 2011-05-14
[27] Google Login PHP Class, http://andrewpeace.com/php-google-login-class.html, 2011-04-20
[28] Designing Interactive Systems: People, Activities, Contexts, Technologies, David Benyon, Addison Wesley, p 64