Project Overview

Audience participation provides an engaging opportunity to dynamically change the focus of a presentation to more effectively meet the presenter's goals, as well as gain valuable insight into the perspectives of those in attendance. In the past, verbal audience polls have been used to get audience feedback, a clumsy process for gathering anything more than rough estimates for yes/no questions.

U-MO and project U-Survey are expediting audience participation using mobile and web technologies to allow the presenter to acquire a lecture hall's worth of input in seconds, and all that's required of the audience is familiarity with their own cell phones. Lecturers will be able to preplan a series of questions prior to presenting or add questions during the presentation, and all responses are collected for later reference.

Similar systems are already in existence, but they all require propriety clicker devices that are very costly for audience members looking to participate. U-Survey will eliminate this need by allowing audience members to leverage the use of their own cell phones and other mobile devices in order to participate.

Project U-Survey will provide presenters with the power to interact with and learn about everyone in attendance during the presentation, providing an enhanced view into the minds of the audience and a greater opportunity to meet its needs. The software will change the dynamic of presentations from a one-way push of information to an exchange of thoughts and ideas where all voices are heard.

Basic Requirements

Registration
Users will be able to register for the system, creating a login name and password used to access
the system. They will also be able to specify if the account should act as a presenter account, allowing them access to various presenter specific features.

Login
Users will be able to login to the system by specifying their account username and password. This will allow them access to the main features of the system.

Forgot Password
Users will be able to retrieve their account password, in case they have forgotten it. They will specify their account email and an email containing instructions for resetting their password will be sent to them.

Survey Creation
The system will provide a way for users with presenter accounts to create new surveys. They will be able to specify a survey name, as well as create any initial question and answer sets they might want.

Survey Management
The system will provide a way for users with presenter accounts to manage existing surveys. They will be able to delete surveys, removing those surveys and all associated questions and answers from the system. Editing of surveys will be possible allowing for editing of associated questions and answers. Questions and answers will be able to be moved up and down relative to other questions and answers, allowing for reordering.

Survey State
Users with presenter accounts will be able to change the state of a survey between two sets of closed or open survey states. The closed survey states are incomplete and closed, and the open survey states are available, private and restricted. Incomplete surveys are those that are defined as still being worked on by presenters and won't be accessible by anyone but that presenter. Closed surveys are surveys that have ended and won't be available to audience members. Presenters will have the ability to re-open closed surveys, allowing audience members to register and take the survey again. Available surveys are those that are defined as being open to everyone to take and register, as well as appearing in search queries. Private surveys are surveys that will only show up in the search if that surveys exact unique id is entered, preventing any unwanted audience members from registering and taking the survey. Restricted surveys are surveys in which a presenter will create an audience list for, and everyone in that audience list will automatically be registered for the survey. Restricted surveys won't show up in the search at all. Presenters will be able to change a new surveys state to one of the three open surveys at will. After a survey has been opened a presenter will be able to close it at will.

Audience Lists
Presenters will be able to create audience lists for a given survey, allowing them to create predefined lists of users of which they believe will take the survey. They will be able to search for a given user by name, view a list of search results, and add any users they wish to the audience list. Users added to the audience list will automatically be registered for the survey.
**Survey Results**
The system will provide a way for presenters to view results of audience responses for surveys. Results can be viewed based on group results for a given survey, or an individual audience members specific responses to one of the surveys they have taken. Group results will display the information in the form of a bar graph, with a graph for each question and the number of responses to each answer plotted on the graph. Individual results will display a list of all questions and answers for the selected survey, with the given users responses highlighted clearly.

**Survey Registration**
The system will provide a way for audience members to register for a survey. They will be able to find a survey through searching for it by either name or unique id. Before registering for the survey they will be able to preview it and cycle through the questions but won't be able to submit answers for any of them. In place of the submit button will be a register button. After the user has registered for the survey, it will show up in any my surveys lists they have.

**Survey Participation**
The system will provide a way for audience members to participate in surveys after registering for them. They will be able to view questions and their associated answers one at a time and submit responses to these questions. These responses will be saved by the system for later retrieval. Audience members will be able to begin submitting responses to surveys as soon as they are registered. Audience members can go back and change their answer to a question at any time, the system will update their response accordingly.

**Constraints**

One of the main constraints on the system in terms of mobile development was the lack of access to a wide variety of mobile device models. The team members personally had access to an Iphone and Ipod touch, as well as other cell phones, but these cell phones did not have data plans so we weren't able to access the project on them. Later in the term the project sponsor gave us access to a Nokia E71 mobile phone, but prior to this, most of our mobile development was focused on the Iphone and Ipod touch. These devices are fairly high end in their web browsing capabilities so it was hard for the team to get a grasp of how the project might look on lower end cell phones. We did discover an online cellphone simulator which we could use for testing, but this was found later in the project and proved to be unreliable at times.

Time was a major restraint on some of the initial requirements. SMS technology was looked at in the beginning of the project, but time and resources prevented us from doing any actual development in this area. The same can be said for the development of Iphone or Nokia Symbian specific applications, we simply didn't have enough time over the two quarters of the project to look into these with any detail.

**Development Process**
The team choose the development process Scrum early on in the project as the process to use for developing the system. This process was approved by the project sponsor, in fact he was already familiar with the workings of Scrum, so it was a good fit. The process was mandated to the
extent that there were two weekly meetings held with the faculty coach, who acted as the scrum master. The faculty coach also made sure we were following the process in that all the necessary documents and artifacts were updated as needed.

Communications with the sponsor occurred once a week in the form of a sit down telephone conference meeting. At the beginning of the project, the product backlog was created and discussed with the project sponsor, using a System Requirements & Specifications document to identify features. He looked over the items and gave each one a priority ranking, used by us to determine what to focus our development efforts on first. At the start of a sprint, we would tell the sponsor what we planned to accomplish for that sprint, and he would either give it the ok or tell us what needed to be revised. In this way, the sponsor was the one who determined what got developed and when. Meetings with the sponsor generally consisted of us telling him the progress we had made for that sprint and asking him if he had any questions for us.

The roles the team identified were mostly mandated by the Scrum process itself and included Scrum Master, Product Owner and Scrum Team. The Scrum Master was our faculty coach, Michael Lutz, the Product Owner was our project sponsor, Dean Andrews and the Scrum Team was the four of us.

**Project Schedule: Planned and Actual**

The project schedule followed more of an evolving and adaptive style consistent with the Scrum process. We didn't explicitly define the entire project schedule at the outset, but instead contained it to each 2 week sprint as a whole. Because of this, the project plan at the beginning cannot really be compared to the state of the project at the end. At the beginning of a sprint, the items to be worked on were chosen from the product backlog and put into the sprint backlog. This was looked over by the project sponsor and any necessary modifications would be made from there. This allowed for easy changes to the schedule in case anything should come up. Since each sprint was confined to 2 weeks of development, if anything drastically changed priority we could just put that into the next sprint fairly quickly and easily.

**System Design**

Our system design followed an evolution similar to the rest of the project as features and technologies changed and grew. Early on we settled on PHP as our web programming language and MySQL as our database language. In the beginning of the project we initially started developing using just these technologies, making everything ourselves from scratch. Part way through the second sprint our faculty coach mentioned it might be useful to look into PHP frameworks that supported a Data Abstraction Layer (DAL) in case we decided to change our database technology down the road. After investigating a few other frameworks we eventually settled on the CakePHP framework. We felt that CakePHP seemed to have the most support, was the easiest to get up and running, provided a useful set of core functions that would make developing easier and provided the DAL we were looking for. After the switch to using CakePHP we had to refactor our previous work from the first sprint. This was made easier by the fact that much of what we had done for the first sprint was front-end html code, which was easily transferable to CakePHP. CakePHP proved to have a bit more of a learning curve than we
anticipated, which led to some slow development at first as we learned the ins and outs of the framework. At times, it did seem like we were fighting against CakePHP in order to get the desired functionality, but overall it proved more of a help than a hindrance.

By using CakePHP, much of our system design was already laid out for us, we only needed to follow CakePHP's predefined standards. Therefore, our system design matches that of CakePHP's architecture. Since we were using the predefined framework of CakePHP it was not necessary to consider modifying the existing architecture which was in place. This would have consumed too many resources and would not have been worth it in the long run as CakePHP's architecture has been tested and verified by many developers. As such, the core architecture of CakePHP is shown in the following diagram.

**Figure: CakePHP Architecture**

CakePHP follows the standard Model View Controller (MVC) architecture pattern. Data is stored and accessed from the models, passed to the controllers where the business logic is applied and finally any necessary views are rendered in order to display information to the end user.

In CakePHP each model class is directly linked to a database table of the same name, and if named correctly, will handle much of the simple create and update calls for you. We made heavy use of certain model relationship types, such as has one, or has and belongs to many, in
order to relate different models to each other. For example, Survey had a has and belongs to many relationship with Question, signifying that a single Survey has a set of questions that it owns.

The system also contains a total of seven controllers that interact with the models to create dynamic content. The two most notable ones are SurveysController and UsersController. These two controllers are the largest controllers in the system as they contain most of the business logic of the system. UsersController is mostly concerned with actions that all users typically perform, such as register, login, logout, and rendering the correct portal pages. On the other hand, SurveysController is typically responsible for the business logic concerned with surveys, such as rendering survey's contents, creating and deleting surveys, adding and deleting questions and answers to surveys, as well as changing the type of surveys.

The views in the system are presentational pieces used to display information to the user. Controllers will pass any necessary data to the view and the view will determine how to render it. Each controller action is associated with a view of the same name, helping to keep a high cohesion of actions. We also made use of view elements, which are reusable pieces of modular presentational logic that can easily be placed in any main view file. For example, we had elements for common search forms, or for common list displays used in multiple places throughout the site.

Figure: Model Structure
This is the basic structure of the models used in the system. This structure helps to determine how things are mapped into the database and what join tables are needed. Every user belongs to either the Audience group or the Presenter group. Each group has a different set of distinct permissions on which actions they may perform. User groups are arranged in a hierarchical manner. As such, Presenters have access to all of the permissions of the Audience group in addition to permissions exclusive to the Presenter group. Audience members have access to view and take surveys as well as create responses, which simply represent the particular answers a user has picked for a specific question in a survey. Members of the Presenter group have access to all of the privileges of the Audience group as well as creating new surveys, questions, answers, and audience lists for those surveys respectively. A survey consists of questions and answers and each survey also has a type associated with it. These types are Available, Restricted, Closed, and Incomplete.

**Process and Product Metrics**

Metrics tracked for the project were defects and tasks. Defects were tracked using a Defect Removal Effectiveness chart, which shows the rate at which bugs are found and removed from the system and helped the team identify which defect removal methods were most effective, as well as identify problem areas that may need to be redesigned. Tasks were tracked using Burndown/Slippage charts, which monitored tasks that were on time with schedule and those tasks that were behind schedule, and helped the team stay on track with the project as well as indicate tasks that need to be further broken down into smaller tasks. The tasks and defects were recorded regularly but the charts were not created on a consistent basis. Creating the charts for both metrics on a more regular basis would have helped the team identify more effective strategies. Overall the results of our metrics showed that the product was on schedule and all tasks were completed that were promised and all bugs were resolved.

**Product State at Time of Delivery**

Right now the product is fully functional and operational with a solid core set of features integrated. All the basic functionality is in place allowing for presenters to create surveys, and audience members to search and take those surveys. A little more polish in terms of usability and minor bug fixes is needed, but other than that the product works as intended. As mentioned before, we didn't have enough time to fully test multiple mobile devices, so the product doesn't perform as optimally as we would have originally planned on these devices.

In terms of missing features, SMS support was planned at the beginning of the project, but is not currently integrated into the product because of time constraints. We had planned to allow audience members without a data plan the ability to submit answers to questions by just sending a text message to the system, but this is not supported in the delivered product. Iphone and Nokia Symbian applications were also planned at the outset of the project if time allowed, but we ended up not having enough time needed to begin developing these. Support for the OpenID login system was also talked about at the beginning of the project as a useful feature. We looked into technologies that supported it briefly, but instead focused our efforts on getting an internal login system in place. As the project progressed we thought our efforts would be better placed in
developing new features instead of trying to support OpenID. The project sponsor agreed with this so OpenID ended up not being supported in the delivered product.

**Project Reflection**

The decisions made by the team throughout the project were the result of appropriately applied risk mitigation techniques. Our selected process allowed us to begin development early in the project, reassuring both the team and the sponsor that production was progressing and would continue to progress. This also afforded us more time and experience with the technology and each other, increasing our development proficiency while timing constraints were still light.

Though the team does feel our technology selection enabled us to meet the needs of the sponsor, in retrospect CakePHP can be seen as limiting due to implementation quirks and complexities. The team feels that another technology exists that may have made simplified development for both our team and future developers, though we still feel the selection was necessary to make given the amount of time we could justifiably spend on research. Again, wise process selection allowed us to acclimate to the technology sooner, and help each other become better developers with it.

**References**