|  |  |
| --- | --- |
|  | **2011** |
|  | APSCUF-KU  Kenneth Rohlfing, Adam Blank |

|  |
| --- |
| **[Progress Report]** |
| Team A's Progress Report up until 11/18/2011 |

v2.2

Submit To: Dr. Tan

Phase 1

# Introduction

This document will serve as a progress report for Team A from the start of the iVOTE system until 9/30/2011. The document will be split into Planning Completed, Design Decisions, and Current Implementation.

# Planning Completed

There were several planning tasks that had to be achieved before moving into the design phase of the first increment. We had to decide on a collaboration and team management system, decide what programming languages were best to complete the project, and decide on using the legacy system or not.

## Collaboration and Team Management

For a collaboration and team management system, we first considered using Google Wave. After first logging into Google Wave, there was a message in red text declaring the system to be in deprecation. Upon further research, it was found that the Apache Foundation had adopted this project, and it would soon undergo transition. The Google Wave system was not impressive to begin with, but this deprecation made it unreliable.

After some searching, we found TeamLab. TeamLab is an open-source, ASP.NET driven web-application that handles project management. This includes milestone and task management, discussions, documents, time tracking, managing team members, and a logged chat system. Our team portal is located at https://csc354g1.teamlab.com/.

## Programming Languages

We decided to use the ASP.NET 4.0 framework in order to complete the system. It would be infeasible to make a non-web system, as it is likely that few APSCUF-KU members would be willing to install an application they would use once a year to vote on officers. A web-based solution made the most sense, as all faculty members would have access to this system. ASP.NET has good built-in security in comparison to PHP, and also has the MVC 3.0 framework as an added feature. This would allow us to implement the system in an object-oriented fashion, without worrying as much about security.

## Legacy System

We decided not to use the legacy system. While their documentation was impressive, their code did not work correctly, and they had very little complete. It seems better for us to design our own system and build it from scratch.

# Design Decisions

Most of the time we spent up until this point has been focused on the design of the system, and breaking down that design into a small section for the first increment. We spent a great deal of time debating the procedures document, and how to best implement these requirements in a safe and secure way that could not be abused or was flawed logically. After analysis of this procedures document, along with a client meeting with Karen Epting, we were able to come up with a list of requirements for the system. From this list of requirements, we began laying out how the system would be organized in order to achieve these goals. Once this high level abstraction of the system was complete, we began work on lower-level design for the first increment.

## Procedures Document

During the debate over the procedures document, we were conflicted on several points. First, we had to discover the scope of the project. This was clarified by Dr. Tan to be only section one of the procedures document, as Karen Epting only expected the system to handle the elections of Officers and Delegates.

Second, we had to decide the best way to distribute willingness-to-serve and nomination forms. We decided that creating web-based forms would be the best way of doing this. Using fillable PDF forms are expensive in terms of developer time and faulty. Distributing and collecting paper forms are expensive in terms of time, and the results would need to be manually entered into the system. A web form requires little development time, is low-risk in terms of failure, and can automate the data handling process.

There was some confusion over plurality elections in comparison to majority. After some research, it was found that plurality is an iterative election style, where each round of voting elects a candidate to a position. This allows the overall voter opinion to be better represented. Majority voting is capturing over 50% of the vote. If no candidate captures over 50% of the vote, the candidate with the least votes is dropped from the ballot and a revote is issued. These notions were confirmed by Karen Epting.

## Safety and Security of the System

Certain considerations had to be taken in order to ensure the system would be secure and trustable. Each user must have an account secured by a password, and accessing functionality would require being logged into this account. There also had to be several account roles with different levels of functionality.

Initially we planned to have a registration form for users to register their own accounts. Karen Epting would have to verify that they were APSCUF-KU members, and then they would be given access to the system. However, during an interview with Karen Epting, she requested that she create the accounts by using a web-form. She stated that there are, on average, four new members per year, and this would be easy for her.

Therefore, a requirement of the system was a user creation form for Administrators. After an Administrator enters in a new user account, an email is generated and sent to the user's email address. This email includes an activation link that will allow them to create a password for the system. This password is expected to be known only by the user. This ensures that only they may log into their account, and that it is them placing any votes, nominations, or filling out any forms.

There are needs for different levels of user accounts. The basic level of account is "Faculty." All other account roles inherit their basic privileges from this role. This role would be able to log in, log out, create a password, change a password, recover a password, nominate, fill out willingness-to-serve, petition a nomination, and vote in an election.

The Nomination and Elections Committee role would have all of the privileges of the faculty plus the ability to view election results, approve nominations, and approve votes.

The APSCUF-KU President role would have all of the privileges of the faculty plus the ability to view and publish election results.

The Administrator role would have all of the privileges of the faculty plus the ability to create user accounts, edit user accounts, delete user accounts, create elections, view election history, and approve eligibility.

By assigning and maintaining these roles, only users with proper authorization will be carrying out their respective tasks.

## Design for the First Increment

For the first increment, we set out to fulfill several requirements in the system. We wished to give users the ability to log in and log out, the ability for an Administrator to create, edit, and delete users, and for users to be able to create their passwords using the verification email.

Logging in requires that the user have an account with a created password for the verification email. The user will enter their email and password into a form and submit it. If their credentials match the set in the data store, then a session will be created and they shall be logged in. If not, they will be returned to the login form with a corresponding error message.

Logging out requires that the user already be logged in. This event will occur if the user clicks the logout button, or if they are inactive for fifteen minutes.

Creating a user will occur every time a faculty member becomes an APSCUF-KU member, and Karen Epting receives their information. For this increment of the system, only basic information is used, such as first name, last name, email, phone number, and department. When the Administrator submits the form, a new user account will be created and a password creation email will be sent to the user.

Editing a user will occur if the data on record is incorrect or updated. The Administrator will search through user accounts to find the record to be edited, and the information shall be loaded into a form. After editing the desired information, the Administrator will submit the form and the record will be updated.

Deleting a record will occur if a faculty member decides to leave the union. The Administrator will search through user accounts to find the record to be deleted, and then confirm that the information is to be deleted. After the form is submitted, the user record will be destroyed.

The password creation email will be sent on the creation of an account. The user must enter in the password twice, once as the desired password and a second time to confirm it. After the user submits the form, their password will be set and they shall be able to login to the system.

# Current Implementation

The current system in place consists of a generic testing website hosted on our development server, with the account creation process fully implemented. The current system is database centric, using MySQL as the database management system.

The account creation system is fully implemented and consists of two end-user pages. The first page consists of a form that the Administrator of the site may fill out. This form has five input fields: First Name, Last Name, Email, Phone Number, and Department. These fields are current checked using ASP.NET's "Required Field Validator" and "Regular Expression Validator." The regular expression validators ensure that the email address registered is either a @kutztown.edu or @live.kutztown.edu account, and that the phone number field is numeric.

After the form is submitted and successfully validated, several events occur. First, two unique 64-byte strings are generated using a custom, regular-expression based password generator. Next, the new user's information is added to the appropriate table in the database, and the index key value of the newly added row is returned. The newly created index key is stored with the two 64-byte strings in another database table. An email is then sent to the email address provided by the Administrator on account creation, using the test server's SMTP server.

This email includes a message letting the new user know that their account has been created, and provides a link to the account confirmation page. This link has a URL query containing one of the generated codes to confirm. There is another link with a URL query containing the other generated code to invalidate the account if the email was sent to the incorrect person.

On the confirmation page, a check is performed to see if the URL query matches the 64-byte string stored in the database. If the code is correct, the user record is fetched. The user is prompted to enter their desired password twice. The repeat is to ensure that they did not make a typing error while entering the password for the first time.

When the user submits the form the passwords are checked against each other using an ASP.NET validator. The database record for the user account will have its password field updated, and the account will be considered validated. The account will be able to log into the system, which is currently being worked on. The log in system, at the creation time of this progress report, is what the development team is working on.

Phase 2 Sprint 1

# Introduction

During this phase, we set into a weekly developmental cycle. Every Wednesday, we held our weekly meetings. These weekly meetings were the planning for the upcoming week's prototype. After this meeting, James would make the use-case descriptions for that implementation. On Thursdays, Aaron and Ken would meet to complete the system sequence diagrams. On Fridays, James and Ken would meet and do the database design. This would be after the team testing sessions, where our previous planning cycle had to be implemented. During the course of the upcoming week, the planned increment would need to be implemented. Also during this time, test cases had to be created, which were based off of the use-case descriptions created by James. At the end of the week, system testing would occur. Any errors found in the system would be added to the MR report, and would be corrected by the developers.

For this phase, I will elaborate what happened during each of the three weeks of development. This will include the planning, design, implementation, errors, and revisions.

# Week 1

## Planning

For this week, we laid out the requirements by role as set out in the SRS document. We planned to implement the administrator's approve eligibility for willingness to serve forms, faculty's change password, and the recover password functionality.

The approve eligibility form was planned to be a simple form based off willingness to serve data in the database. Since willingness to serve forms were not yet implemented, it was planned that we would manually fill the database to test this.

The change password form was planned as a simple form where a logged in user would enter their current password, the new password, and then confirm their new password. The current password would be checked against the database, and the new passwords would be checked with an ASP.NET compare validator.

The recover password form was a reuse of the create password form, using the 64-byte validation codes. The user would simply enter their email address, and if the address exists in the system, they would receive an email leading them to the create password form.

## Design

During the design phase, James made the use-case descriptions for approve eligibility, change password, and recover password. From these descriptions, Rebecca formed the test cases.

Aaron and Ken create sequence diagrams for the Willingness to Serve process, which included the approve eligibility process. They also created sequence diagrams for change password and recover password.

James and Ken met to design the willingness to serve table for the database. This table needed to store the user's id number as a foreign key, the position they were willing to serve for, and the statement they prepared for that position. We decided to set the limit on their statement to 1024 characters, as that is the average number of characters on a page of text.

## Implementation

During the implementation phase, the developers worked to implement the approve eligibility form, change password form, and recover password form. Tommy failed to create the change password form after waiting to the night before, sent an email at 1a.m. the day it was due, and Ken had to rush to push out a change password form before the system testing for that week. At this point, Tommy was warned that if he does not contribute more to development, he would be fired by the end of Phase 2 Sprint 1.

Towards the end of the implementation cycle, Dr. Tan issued requirements to be done for that test prototype. We had to rush design and implementation for the ability to sort and search for users.

## Errors

There were several minor errors found during system testing for this week. The found that two people can log in with the same information at the same time. We decided that this was not an error or an issue, as account security lies with the individual. Also, the user may be signed on with a mobile device and a desktop device at the same time, causing concurrency issues. It was found that some users did not receive a confirmation email on the creation of users. They were able to find these emails in their spam folders. It was requested that the change password functionality be added to the navigation, as the link had to be typed into the address bar. This was due to the change password form being developed last minute by Ken. On the change password form, the ASP.NET compare validator would say that the passwords do not match until the user clicked out of the boxes. On the recover password page, you are able to enter an invalid email address, or an email address that does not exist in the database. We originally intended it to be this way, to prevent anyone from trying to phish email addresses in the system. However we did take action on this. On the change password page, after successfully submitting the form, the confirmation label was set to the username. This was done for testing purposes, but Ken forgot to change it back to the confirmation message. For some search strings, the user was not able to find who they were looking for.

When editing a user, the other team was able to edit a user to have the same email address as another user. This was a more severe error.

## Revisions

All of the revisions were assigned to Tommy, as they were seen as easy modifications, and it would give him a chance to participate more with development. He was able to update the confirmation message on the change password form, although he did not do so correctly. He was not able to add the change password functionality to the navigation, Ken and Ralph completed that task the morning of the next system test. He was not able to update the edit user form, changing the email textbox to a label so it could not be edited. Adam completed this task. He updated the ordering of the inputs in the change password form such that the ASP.NET compare validator would update before the user hit submit. Adam made the recover password form check to see if the email address existed in the database, and prompted the user if it did not.

# Week 2

## Planning

During this week's meeting, we primarily focused on the user interface design. This topic was brought up by Adam, who led a majority of the meeting. He brought up the point of making the system for the user, who would only being using the system for short periods of time, and who did not want to think while using it. Our system should end as a single page with no menu, but instead be alert/notification driven. The page would only display functionality related to the current period of the election. Ken expressed concerns with some of the ideas, as they may break ADA compliance, but that would have to be tested during development.

We planned to implement the willingness to serve form for the upcoming system test.

## Design

During the design phase, James made the use-case descriptions for willingness to serve. From these descriptions, Rebecca formed the test cases.

Aaron and Ken modified the sequence diagrams for the Willingness to Serve process, breaking it into willingness to serve and approve eligibility.

Jeremy created an initial draft of the user interface over the course of this week.

## Implementation

The developers implemented the willingness to serve form in such a way that when you fill out your willingness to serve, you are nominating yourself. This closely ties the nominations and willingness to serve features.

## Errors

Only minor errors were found this week. One error was that the willingness to serve could be submitted multiple times for the same position. Also, a typo was found on the willingness to serve form.

## Revisions

Revisions were assigned to Tommy. He was able to fix the typo. He broke the willingness to serve form trying to fix it. Adam and Ralph fixed the willingness to serve form.

# Week 3

## Planning

During this week's meeting, we planned for the nomination process, the petition process, the NEC nomination approval, and the acceptance or denial of nominations.

For the nomination process, the user would be able to nominate themselves or other users. The nominations for other users would be search based, as compared to listing all users. This would be infeasible and annoying when the system has over 400 users. The nominated would then need to accept or decline the nomination. In order to accept, the user must fill out their willingness to serve form. Their nomination will be considered pending until they have their eligibility approved and have the nomination approved by the NEC.

The petition nomination process is similar to the nomination process in many aspects, except that 10 users must place a petition for a user before they are nominated.

## Design

During the design phase, James made the use-case descriptions for nominate self, nominate other faculty, accept nomination, and petition nomination. From these descriptions, Rebecca formed the test cases.

Aaron and Ken create sequence diagrams for the nomination process, the petition process, the NEC nomination approval, and the acceptance/rejection of nominations.

James, Ken, and Adam met to alter the willingness to serve table to better support the nomination process. The table needed additional fields including NEC\_approved and nomination\_accepted. We also designed petition tables to handle the petition process.

Jeremy worked on a second draft of the user interface over the course of this week.

## Implementation

Adam created the nomination form and the petition form. Ralph created the accept nomination form, which was lost due to an ARVIXE data center crash. Ralph worked quickly the morning of the presentation to recreate the form. Tommy failed to create the nomination approval page for the NEC.

## Errors

Due to issues with ARVIXE hosting, some code for this week was lost and the database server is not fully functional. We will need to redevelop some code to make up for this.

Phase 2 Sprint 2

# Introduction

This sprint started off with our team deciding to fire Tommy Kauffman. Firing Tommy was the start of reworking the team and our processes. In order to help bolster development, Kenneth Rohlfing began assisting with development as he could. We decided that Skype would be our means of instant message communication, as it has chat log history and both voice and video calling. Team members also had a difficult time understanding TeamLab, and were not motivated to use it. We decided to create a Google Site to manage tasks and our modification request record. We decided for our final presentation that we would use Prezi instead of PowerPoint, as the concept of PowerPoint is old and does not help to keep attention on the presentation.

We no longer followed strict XP development customs. The realization was made that perfecting a deliverable every week would damage our overall progress and time, as we would have to keep reorganizing components when we really did not need to. We focused less on perfection for the testing sessions, and more on the big picture of having everything in running order for the acceptance testing on November 30.

# Planning

This sprint was not heavily focused on planning. Much of the planning for the project had been previously completed, and continuing to focus most of our meeting time on it was spending time we could be utilizing elsewhere.

Early in the phase, we planned out several objects that would be the heart of the system. We planned the election object, the timeline object, and we planned the view object. The election object was meant to handle all data regarding an election, and to communicate with the view to render the correct UI for the user based on their role and the current election phase. The timeline object was planned to handle the phasing of elections by storing the dates of each phase. Every election object would contain a timeline. The view object focuses on rendering a minimalistic, simple web page for the user based off of their needs for that time period in the election. This object will be discussed in more detail in the UI design section.

A developer meeting was held to discuss the progress of the system, database design, remaining work to be done, and to straighten out any misconceptions in the design. The meeting was very insightful and constructive in comparison to the full group meetings, as we did not have interjections or tangents to chase. Everything was very focused and detailed.

After that developer meeting, we have been planning our final presentation and tweaks to the UI. Our resources have been more dedicated to the UI development, and implementation of remaining functional requirements.

# UI Design

Most of the UI design was driven by the view concept. Every page was broken down into three simple parts: Notifications, a Text Body, and Functionality. The administrator role has an extra section just for administrative functions. These parts are rendered based off of the user's role, and the current phase of the election. There election phases are as follows: Stateless, Nomination, Slate Approval and Acceptance, Petition, Approval and Acceptance, Voting, Ballot Approval, and Results. Each of the phases are explained as follows.

## Stateless

The stateless period occurs when there are no active election phases. This could mean that there are no elections at all, or that the role has no functionality in the current phase. This page has no notifications or phase functionality. For faculty and NEC, only a simple text body is shown stating that there is nothing for them to do at this time. For administrator, the only time they will see this page is when there is no election. Therefore, they have all of their special functionality at disposal including election creation.

## Nomination

The nomination period is the first phase in an election. There are two types of notifications for this phase. The first type is visible to any type of user role: pending nomination. This will allow a user to go to the accept/deny nomination page. The second type of notification is administrator only: pending eligibility. This will appear when there is new eligibility to be approved. The nominate functionality will appear on the page, allowing a user to nominate themselves or another faculty member.

## Slate Approval and Acceptance

# Errors/Issues