The Student Scholarship and Creative Achievement Day Application Software

By: Charles Alan Hofer

Introduction

Bemidji State University started the Student Scholarship and Creative Achievement (SSCA) Conference in 1999 to showcase the accomplishments of students. Originally students applied by filling out a paper copy of the application, eventually a secretary would type the information on each application into a word processor and from there information was extracted into a database program. This of course can lead to transcription errors and is occasionally time consuming.

To streamline this process a program was written that enables students to apply for the presentation online. This saved a lot of time and money for the university. The program was designed to be a solution for a single year and could be re-written for the following years.

The original SSCA software was redesigned yearly to accommodate new information. The cause of this was that the original design was set up so that it would work. While every program does need to be updated, the work that necessarily needed to be done took time away from improving its quality. The program had been hard coded so that the information that changed on a yearly basis is scattered throughout the code. Hard coded is a term that is used when data that is consistently changed in several different areas is embedded within the software. The ideal software has all of these parts pulled out and placed together so that updates are made easily. This data, such as the information gathered about the students, would have to be checked every year to make sure that it was up to date in every function. The person doing the updates would need to have some programming knowledge to make some of the simplest changes.

The software package that I have created allows the site administrator to make adjustments to the data that is changed yearly. The first component of the software gathers from the administrator the new data for the web pages and the generated application forms. The second component of the software is the generation of the student application forms. There are several pieces that operate, based on the students' application, to verify and submit the application to the administration. Allowing the data to be updated easily for both parts ensures that the software can be updated in future years, which necessarily increase its capabilities.

Starting the Project

Dr. Marty Wolf, the creator of the original software, discussed with me the possibility of working on this project. The project would entail updating the software package that they were using so that it would no longer be hard coded; would therefore be easier for them to update in the future. The original program had been written in Python, a programming language that I used before in a previous class. Since I knew the basics of the language, I quickly saw what had to be redone so that the program would still have all of its capabilities as well as the modifications that were being asked for. At this point I informed Dr. Wolf that I would do the project for the university and for my honors thesis.

The first step was to get a basic design scheme together to use for the software. This allowed me to visualize how I would implement my portion of the software so that the setup data would be centralized and accessible in other parts of the software. I knew that eventually I would change parts of the design scheme, since I didn't have an exact idea at this point of how I was going to be storing the data to be reused. After I had this rough design scheme of the project written up, I began to write the software.

Choosing a Language

I chose to write the software in Python. This came about for several reasons. The first reason was, as stated earlier, that I had used Python in a previous class. Secondly, the language is very descriptive and easy to use. Also, Python contains pre-made classes that were useful to this project. And lastly, the older versions of the software were also written using Python.

When I was starting my Computer Science degree here at BSU, Python was one of the first languages that I used. The languages' intuitiveness allows for it to be learned quite easily. Instead of using confusing terms like "cout" used in C++, Python uses a simple "print" statement for output. This example of simplicity is one of the strengths of Python. Python is still under development and has its disadvantages, but all high-level languages have drawbacks in the early stages and some still have pitfalls in them. One example of a pitfall in Python is the cumbersome use of 'self' when making classes. The class objects require that the first variable of each of the created methods be 'self'. The 'self' is an annotation used so that the class knows about 'itself' and all of its properties.

Python has features that allow simple statements to do complex jobs, such as the use of "pickling". This may seem contrary to its descriptiveness that I described earlier. Pickling is the condensing of a data structure and placing it in a file along with the data in that structure. Now, when that data needs to be used in a different program, it keeps the data structure that it had when it was placed in the original file without any extra work by the software writer. The software package that I developed uses this feature in several locations in order to transfer data in a streamlined fashion.

The main reason that I chose Python was that the original software package had already been written in Python. Since the software had been written in Python already, I saw no reason to reinvent the wheel for a newer version of the software. Because I already knew how to use the basics of the software and it contained methods that I knew to be useful, I saw no reason to choose another language. It may seem like I took an easy road to reworking the project, but a lot of work still had to be done on the code that was already in place to make my redesigned parts fit together so that the software would run flawlessly and allow for updates to be made with ease.

Structure of the Software

The software package that I created has three separate components: setup, application, and background (Appendix A). The setup component is necessary so that the other parts of the software have the data that is needed to complete the application generation process and create the web pages used for the application process. The application component creates, verifies, and submits an application. In addition there are several systems that run in the background that enable the software to extract the pertinent data and save it to files needed in the development of printed materials that are part of the SSCA conference. The background software allows for the package to move away from being hard coded and keep all of the setup information centralized. The background information is separated into two main parts.

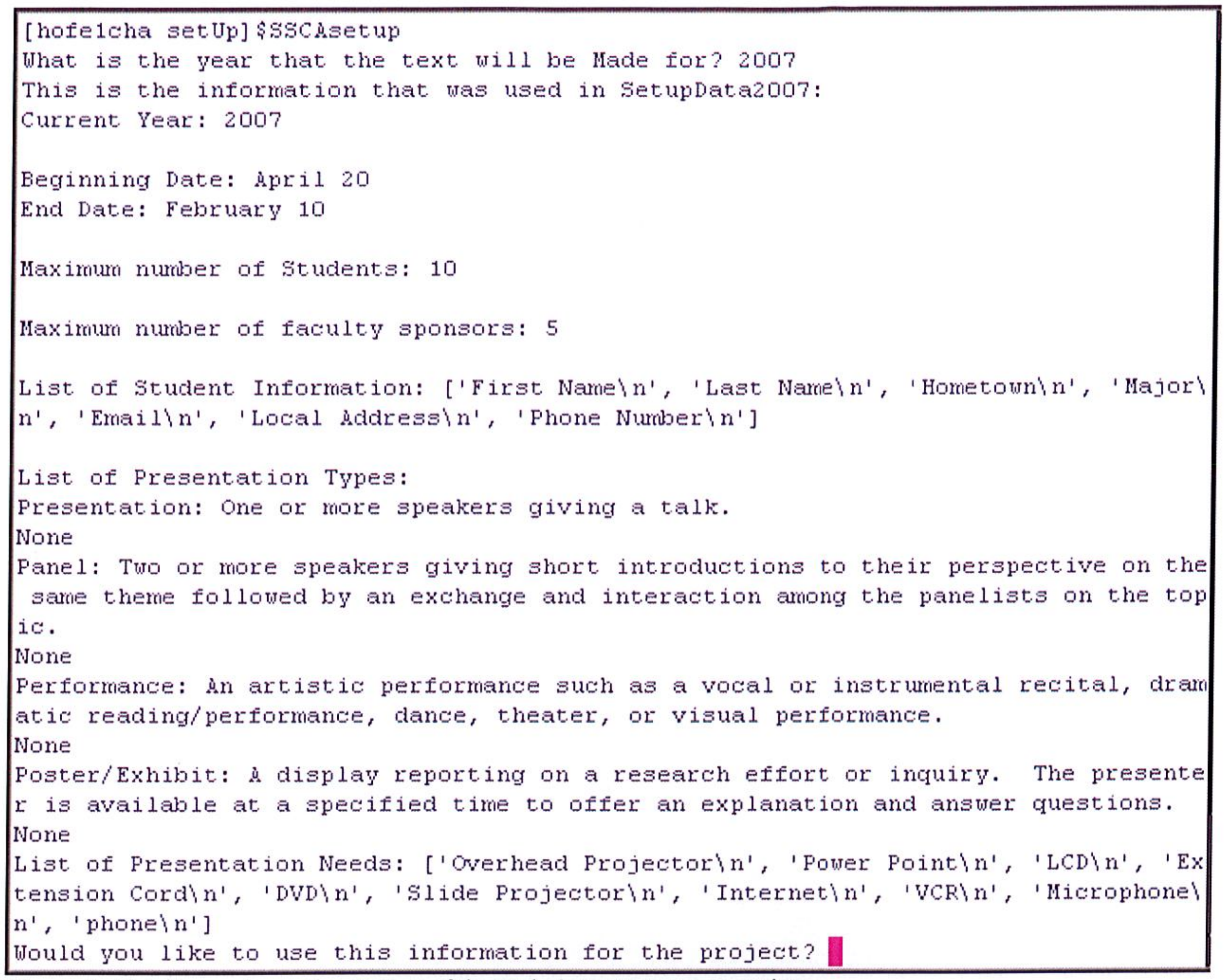
The main part of the background software is the class that is used to grab the data in the setup process. In AppForm.py, a structure is set up at the beginning of the process to hold setup data that can be grabbed and used throughout the software. This includes: the date of the conference, the last day to submit applications, maximum number of students per application, maximum number of faculty sponsors per application, the information that is needed for each student, the types of presentations that are available, and the different equipment that is available for the presentation. Since this data is needed throughout the software, it made sense to create the class structure to contain all of this information. This allows all of the data to be accessed and edited easily. This structure, after getting created at the beginning of the process, is pickled into a file so that the data and its structure are accessible by every part of the software.

The second part of the background component contains several functions and classes that support the setup and application component, as well as the AppForm class. The Helper.py file contains different functions that are used in different places in the software. For example, the header function is used to create the headers for several of the web pages. Two classes are used to help out with creating the setup data. The first, Date.py, is used to check the validity of dates entered by the administrator, such as the date of the conference. This prevents the administrator from accidentally setting the conference date to one that does not exist. The second class, PresentationTypes.py, is a basic structure that gives each presentation entered a name and a corresponding description. Both presentation and description are entered by the administrator. For example:

Presentation type:	Panel
Description of type:	Two or more speakers giving short introductions to their perspective on the same theme followed by an exchange and interaction among the panelists on the topic

This feature allows the Application form to be more flexible, allowing for changes in presentation types and the descriptions that go along with them.

After the background component was completed, the next logical step was the setup component. The first part of this, as mentioned earlier, is gathering the setup data. MakeSetupData.py file (Figure1), makes use of the AppForm class mentioned earlier to create/edit the setup data. This step starts checking if data was setup through this process for either the current year or prior year. The software checks the current year against the year that is



```
[hofelcha setUp]$SSCAsetup
What is the year that the text will be Made for? 2007
This is the information that was used in SetupData2007:
Current Year: 2007

Beginning Date: April 20
End Date: February 10

Maximum number of Students: 10

Maximum number of faculty sponsors: 5

List of Student Information: ['First Name\n', 'Last Name\n', 'Hometown\n', 'Major\
n', 'Email\n', 'Local Address\n', 'Phone Number\n']

List of Presentation Types:
Presentation: One or more speakers giving a talk.
None
Panel: Two or more speakers giving short introductions to their perspective on the
 same theme followed by an exchange and interaction among the panelists on the top
ic.
None
Performance: An artistic performance such as a vocal or instrumental recital, dram
atic reading/performance, dance, theater, or visual performance.
None
Poster/Exhibit: A display reporting on a research effort or inquiry.  The presente
r is available at a specified time to offer an explanation and answer questions.
None
List of Presentation Needs: ['Overhead Projector\n', 'Power Point\n', 'LCD\n', 'Ex
tension Cord\n', 'DVD\n', 'Slide Projector\n', 'Internet\n', 'VCR\n', 'Microphone\
n', 'phone\n']
Would you like to use this information for the project?
```

Figure 1. This figure shows the start up of the MakeSetupData.py execution.

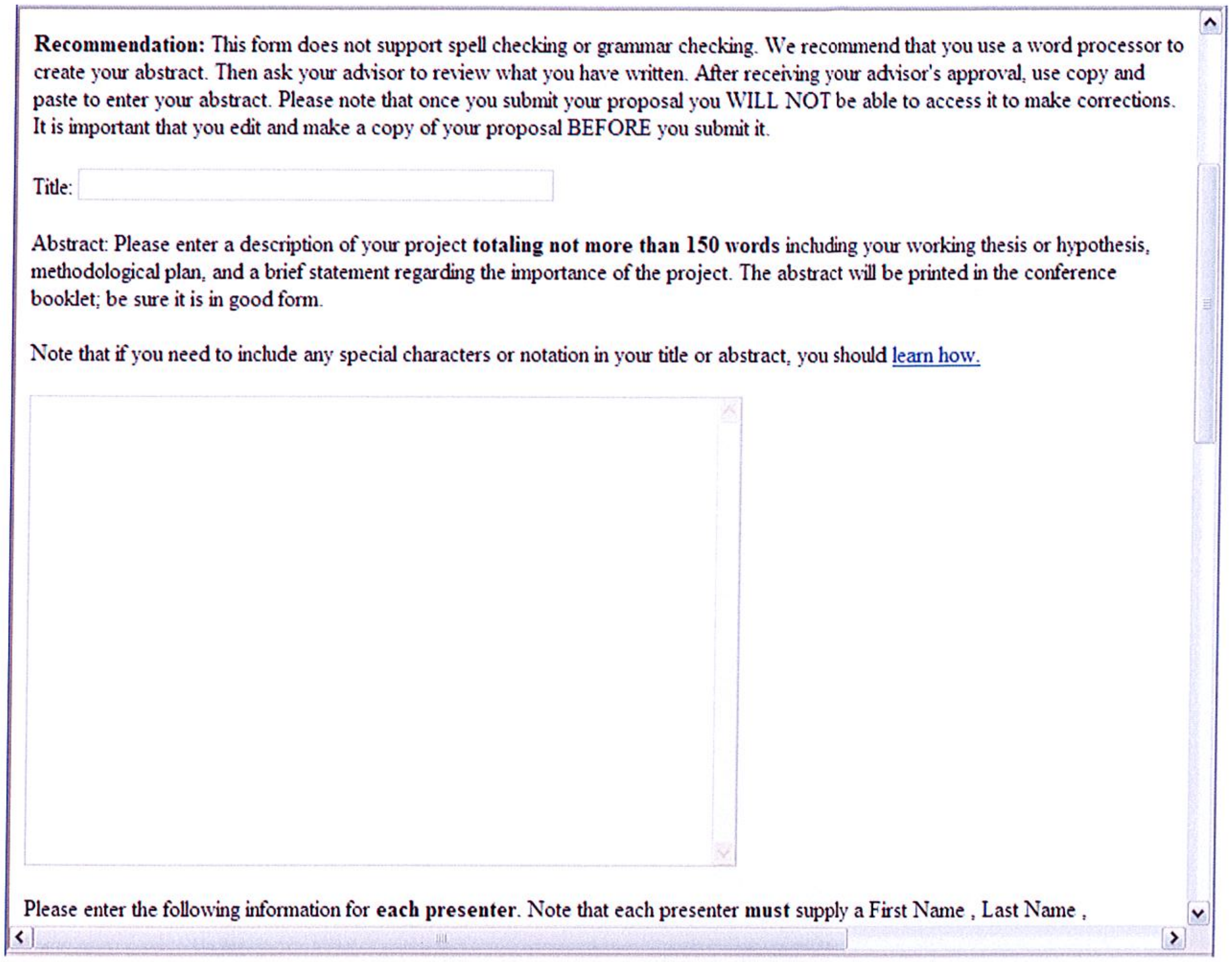
entered by the user. So if the user entered 2007 for the year, the software would check for a data file for 2007 and then 2006. If the 2007 year data exists, the software does not check for the 2006. If either of the data files exists, the user is shown the data and asked if they would like to use the data already gathered. If so, the user is then allowed to edit any of the entries. If the user decides not to use the data file, they are then taken through a series of questions so that a new data file can be populated. After all the new setup data is entered, the user is given the option to edit the information. In the editing process, the user is given a list of options to which part that they would like to change. The list is as follows:

0. Change Nothing
1. Dates
2. Maximum number of Students
3. Maximum number of Faculty
4. Student Information
5. Presentation Types
6. Presentation Needs

After making a selection the software is setup with different ways to edit each of the separate fields. When the administrator is done editing the data, it is then pickled and saved into a new data file.

After the setup data is gathered, the next process in the setup component is creating the HTML files. These files are open.html, select.html, and close.html. They use the data from the setup data file that was created in the previous step. Open.html contains a basic description of the application process, informational dates that are pertinent to the application process, and if the application process is available. It also contains two web links that are useful: a link to the SSCA website and a link to the next step of the application process, select.html. Select.html initiates the application process by having the user select how many students and faculty will be included on the application form. This also contains the link to the application page itself. The other HTML file that is created, close.html is used when the university isn't accepting anymore applications and invites the reader to the conference.

After selecting the number of students and number of sponsors on the select.html page clicking the "Continue with the Application" button runs the next part of the software. Instead of making the application form into an HTML document like the last three pages, the application form is generated by a simple program that gets executed when a button is clicked; the program is generateAppForm.py. This program uses the setup data and creates an application form. The main reason that this form is not created as an HTML document is that the number of students


Recommendation: This form does not support spell checking or grammar checking. We recommend that you use a word processor to create your abstract. Then ask your advisor to review what you have written. After receiving your advisor's approval, use copy and paste to enter your abstract. Please note that once you submit your proposal you WILL NOT be able to access it to make corrections. It is important that you edit and make a copy of your proposal BEFORE you submit it.

Title:

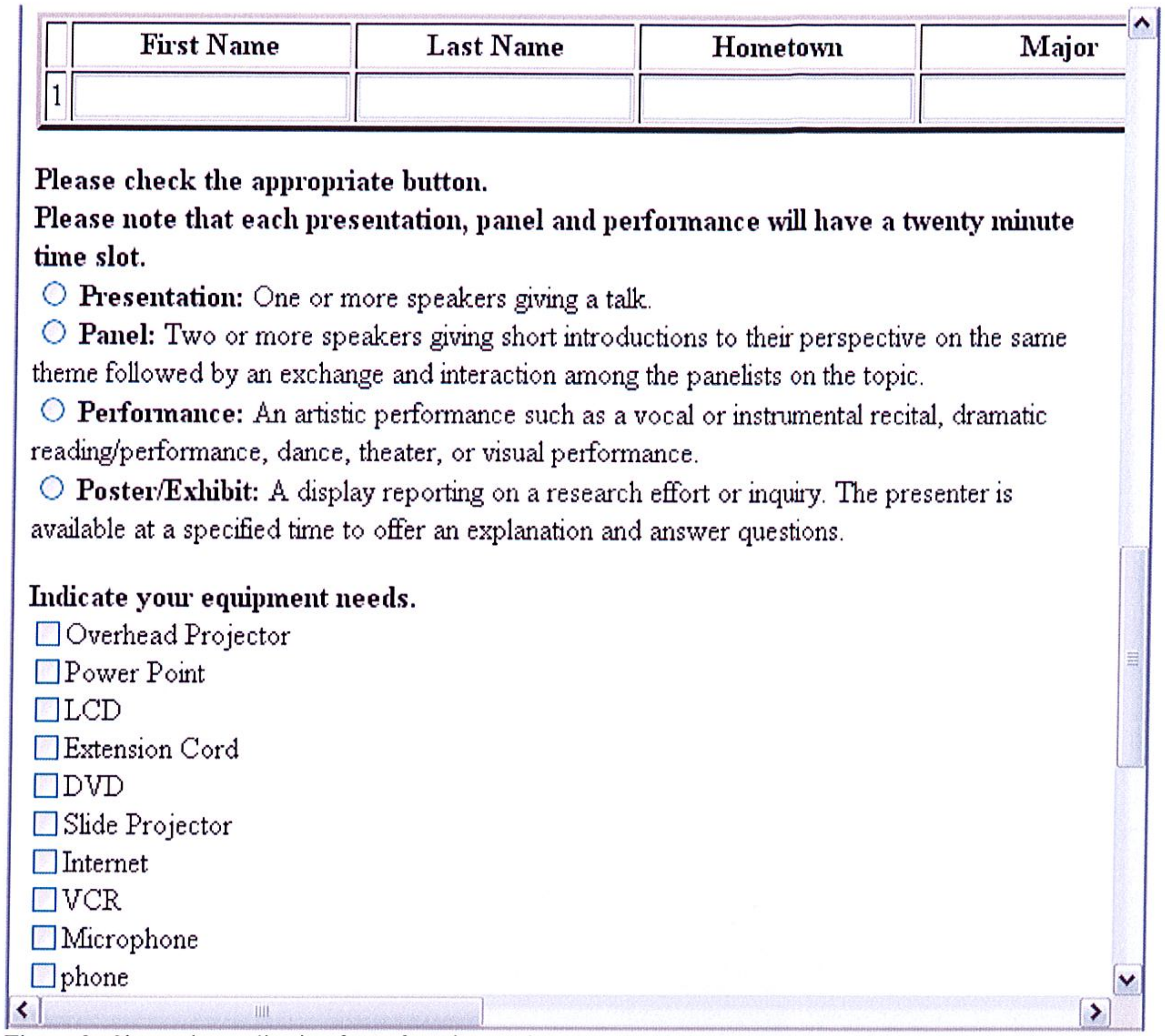
Abstract: Please enter a description of your project **totaling not more than 150 words** including your working thesis or hypothesis, methodological plan, and a brief statement regarding the importance of the project. The abstract will be printed in the conference booklet; be sure it is in good form.

Note that if you need to include any special characters or notation in your title or abstract, you should learn how.

Please enter the following information for **each presenter**. Note that each presenter **must** supply a First Name , Last Name ,

Figure2. Shows the application form, focused on the title and abstract fields.

and sponsors can change for different applications. Generating the forms separately for each application makes the process of checking the application for completion simpler, since the fields are either required or not present on the application. On the application form there are several fields that are not mentioned in the gathering setup data step. These are the title, abstract, special needs and faculty information fields. It is assumed that these will always be needed in the application process. The first field is the title (Figure 2), which is a simple text field. The next field, the abstract (Figure 2), is restricted to 1200 characters. The restriction is used as a means of making sure that the applicant does not write too extensive of an abstract, but instead encourages a more meaningful abstract that is not too lengthy. The type of student information



	First Name	Last Name	Hometown	Major
1				

Please check the appropriate button.
Please note that each presentation, panel and performance will have a twenty minute time slot.

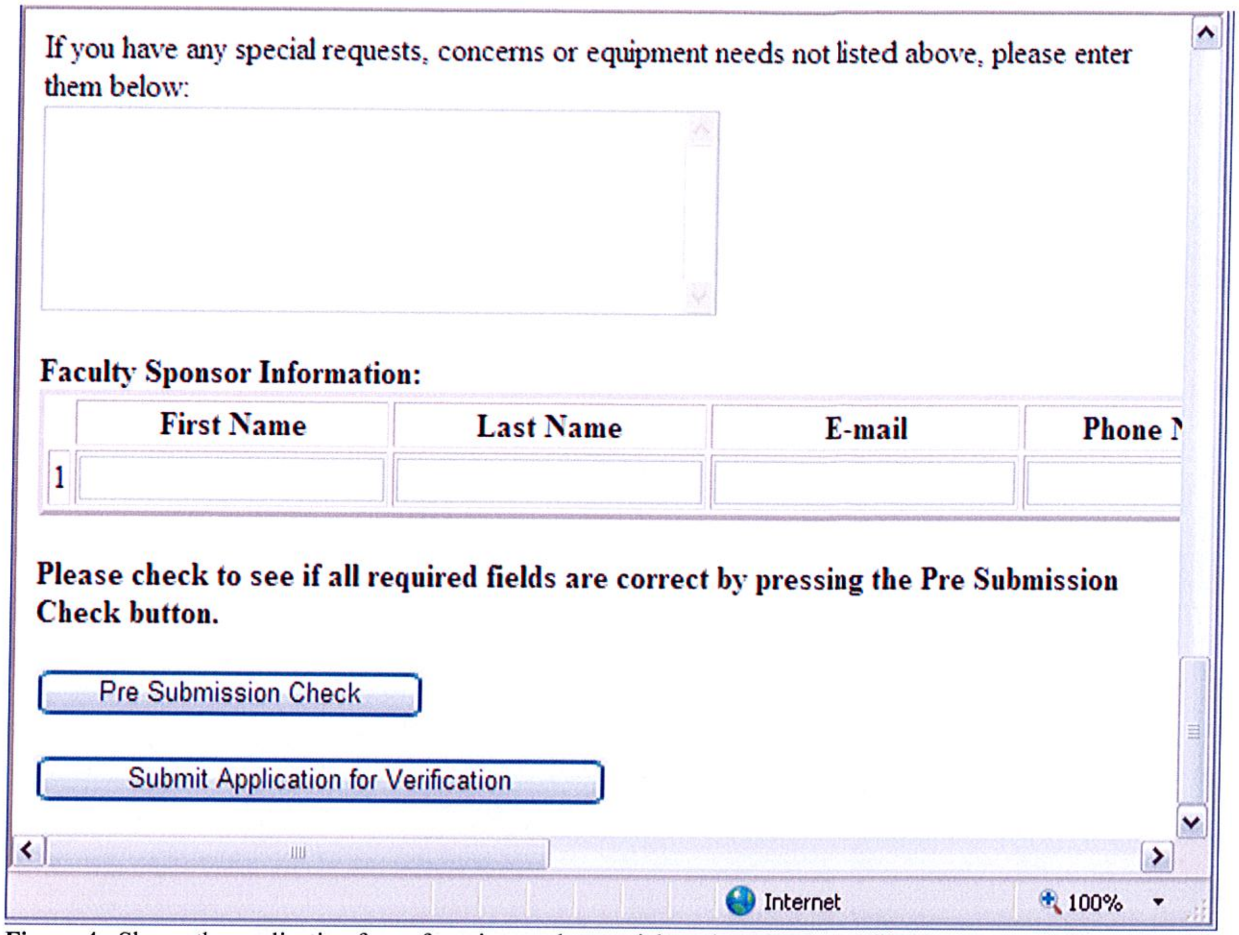
○ **Presentation:** One or more speakers giving a talk.
○ **Panel:** Two or more speakers giving short introductions to their perspective on the same theme followed by an exchange and interaction among the panelists on the topic.
○ **Performance:** An artistic performance such as a vocal or instrumental recital, dramatic reading/performance, dance, theater, or visual performance.
○ **Poster/Exhibit:** A display reporting on a research effort or inquiry. The presenter is available at a specified time to offer an explanation and answer questions.

Indicate your equipment needs.

☐ Overhead Projector
☐ Power Point
☐ LCD
☐ Extension Cord
☐ DVD
☐ Slide Projector
☐ Internet
☐ VCR
☐ Microphone
☐ phone

Figure 3. Shows the application form, focusing on the student table, presentations, and equipment needs

(Figure 3) that was gathered in the setup phase is laid out in a table so that the information can be easily entered. The sponsor field (Figure 4) also uses a table arrangement for the same reasons. It is recommended that students and faculty sponsors use their BSU accounts, because external email sources, such as "Hotmail" seem to block the automated email that is sent out by the software. The most likely reason for this is because of the automated spam blocker that such email providers have initialized. Unfortunately, I did not have enough time to work through the reasons that this is considered spam. The presentations (Figure 3) are setup using radio buttons


If you have any special requests, concerns or equipment needs not listed above, please enter them below:

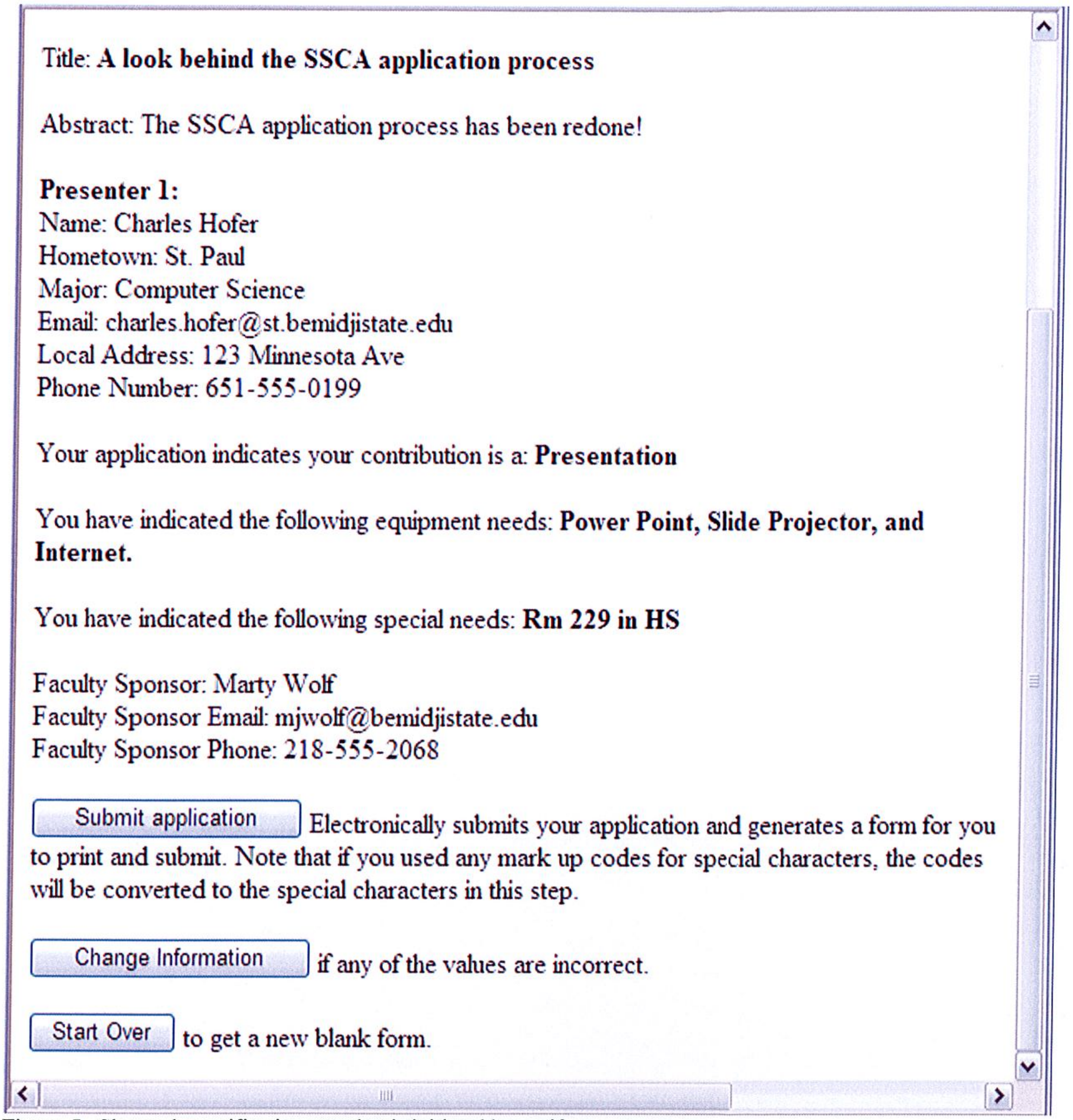
Faculty Sponsor Information:

	First Name	Last Name	E-mail	Phone N
1				

Please check to see if all required fields are correct by pressing the Pre Submission Check button.

Figure 4. Shows the application form, focusing on the special needs and Sponsor fields. Along with the Pre-submission check and submit buttons.

so that one, and only one, can be selected. The equipment (Figure 3) that can be asked for is set up with check boxes since this data is not required for each of the applications. The special needs field (Figure 4) is the only other field that need not be filled in. Each of the fields on the application form is required to be filled, except for the equipment need check boxes and the special needs field. In order to check that the required fields are filled, each is given an ID. IDs allow the software to ensure the fields have a value. This is done using the "pre-submission check" (Figure 4) located at the end of the application. This button accesses a sub program, to check all the fields for values. The Submit Application (Figure 4) is the last part of the form. This of course should only be pressed after the Pre Submission Check is successful.


Title: **A look behind the SSCA application process**

Abstract: The SSCA application process has been redone!

Presenter 1:
Name: Charles Hofer
Hometown: St. Paul
Major: Computer Science
Email: charles.hofer@st.bemidjistate.edu
Local Address: 123 Minnesota Ave
Phone Number: 651-555-0199

Your application indicates your contribution is a: **Presentation**

You have indicated the following equipment needs: **Power Point, Slide Projector, and Internet.**

You have indicated the following special needs: **Rm 229 in HS**

Faculty Sponsor: Marty Wolf
Faculty Sponsor Email: mjwolf@bemidjistate.edu
Faculty Sponsor Phone: 218-555-2068

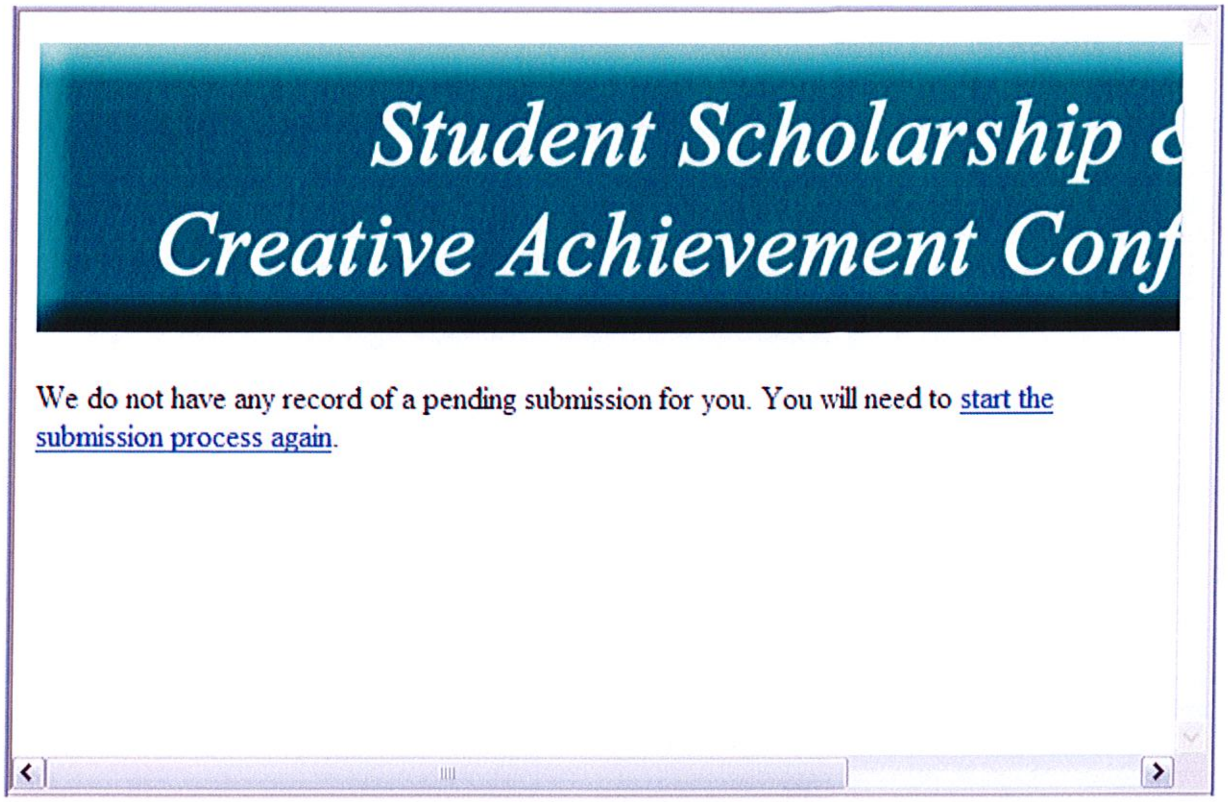
Electronically submits your application and generates a form for you to print and submit. Note that if you used any mark up codes for special characters, the codes will be converted to the special characters in this step.

if any of the values are incorrect.

to get a new blank form.

Figure 5. Shows the verification step that is initiated by verify.py.

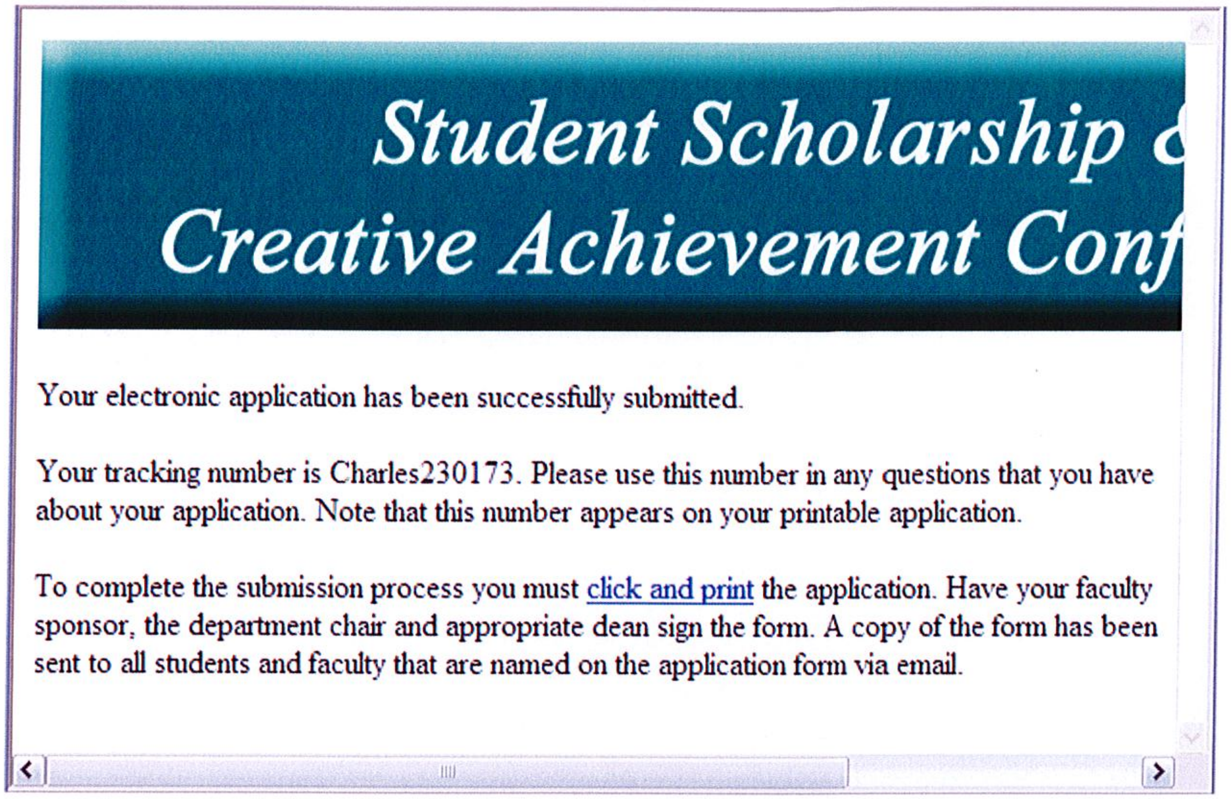
After the user clicks the submission button on the application form, the program calls another program, in the same way as the select.html calls the generateAppForm.py. Verify.py (Figure 5) is designed so that the user can check all of the information that was entered in the application to ensure that it is correct. The submission button continues onto the final part of the application process by pressing the "Submit Application" button. Doing so creates a proposal



We do not have any record of a pending submission for you. You will need to start the submission process again.

Figure 6. Shows the Failed application submission.

file with the first name of the first student on the application and a string of unique numbers (ex. Charles1268364) and executes a program called keepit.py. The page that comes up mentions either that there was an error in the application and therefore the submission was unsuccessful (Figure 6) or the submission of the application was successful (Figure 7). There is also a link present that allows the student to print off the PDF version of the application so that he/she can get the required signatures. Not only is the PDF available through that link, but a copy of it is sent via email to the addresses listed on the application form for the students and faculty members. A Comma-Separated Value (CSV) file is created through this process and used by the administrator. The CSV file is a string of everything that is entered through the application process separated by commas and denoted by double quotes (Example: “Charles”,”Hofer”,”St.



Your electronic application has been successfully submitted.

Your tracking number is Charles230173. Please use this number in any questions that you have about your application. Note that this number appears on your printable application.

To complete the submission process you must click and print the application. Have your faculty sponsor, the department chair and appropriate dean sign the form. A copy of the form has been sent to all students and faculty that are named on the application form via email.

Figure 7. Shows the return of a successful application submission.

Paul", etc). These files are eventually combined into a single file, and the data is then transferred to a database to be used for developing the final program booklet.

<u>What I would have added</u>

There are a couple different things that I would have added to the program if I had more time to work through them. The first is a way to edit previously submitted applications. I also would have liked to add a graphical user interface for gathering the setup data. I also think that I would have added another class used for gathering student information.

Each application that is submitted has a unique tracking number. This tracking number separates the data entered for the application from all the other applications that were entered.

Even two students who apply at the same time for presentations will not receive identical tracking number. The number that is given at the end of the first name is a time stamp in milliseconds that is received from the computers internal clock. To edit a previously submitted application, one would use the tracking number to grab the data that was previously submitted and then populate the application fields for resubmission under the same tracking number.

The interface that currently is used for gathering setup data is just a simple text-based prompt arrangement. I would like to change this, and make it similar to the Application form itself; a simple form with fields. The data that is entered would be easier to see, allowing for the user to have a better idea about the information that is being entered. With the setup that I have currently implemented, it would be easy to modify the program to function the same as the text based interface.

The final change has to do with creating a Student class to keep track of different attributes for the information that is being gathered. One attribute would be the length, which specifies the number of characters that students have to enter for a specific field. A phone number would not need more then 10 characters. One other attribute that I would add is the classification of the type of data that is being entered. In the PDF, the data is separated by two types: student information and contact information. With the class setting up this attribute for the two separate information types being entered, would make the creation of the PDF would be streamlined.

Conclusion

After completing the project I am confident that the software package is set up to the specifications that are needed for the University. Besides the few changes that I mentioned earlier I think that the software will be suitable for use for several years without requiring additional changes. Throughout the project I have consistently experienced what it means to be responsible for a product that is going to be used. All of my class projects have developed for a grade and nothing more. Having a project that is going to be used in real life situations gives me a little bit more respect for those in my field and a better sense of what I will be required to do for my future employer. I have also gained a better sense of what deadlines mean for projects that I am working on. Unfortunately, I was unable to finish my project for the 2007 SSCA application process. I would have liked to have seen my project in use and gotten feedback from those who used it, so that I could have improved it before my time at BSU was done.

References

Python 2.1 Bible, By Dave Brueck and Stephen Tanner. Copyright 2001, Hungry Minds, Inc.

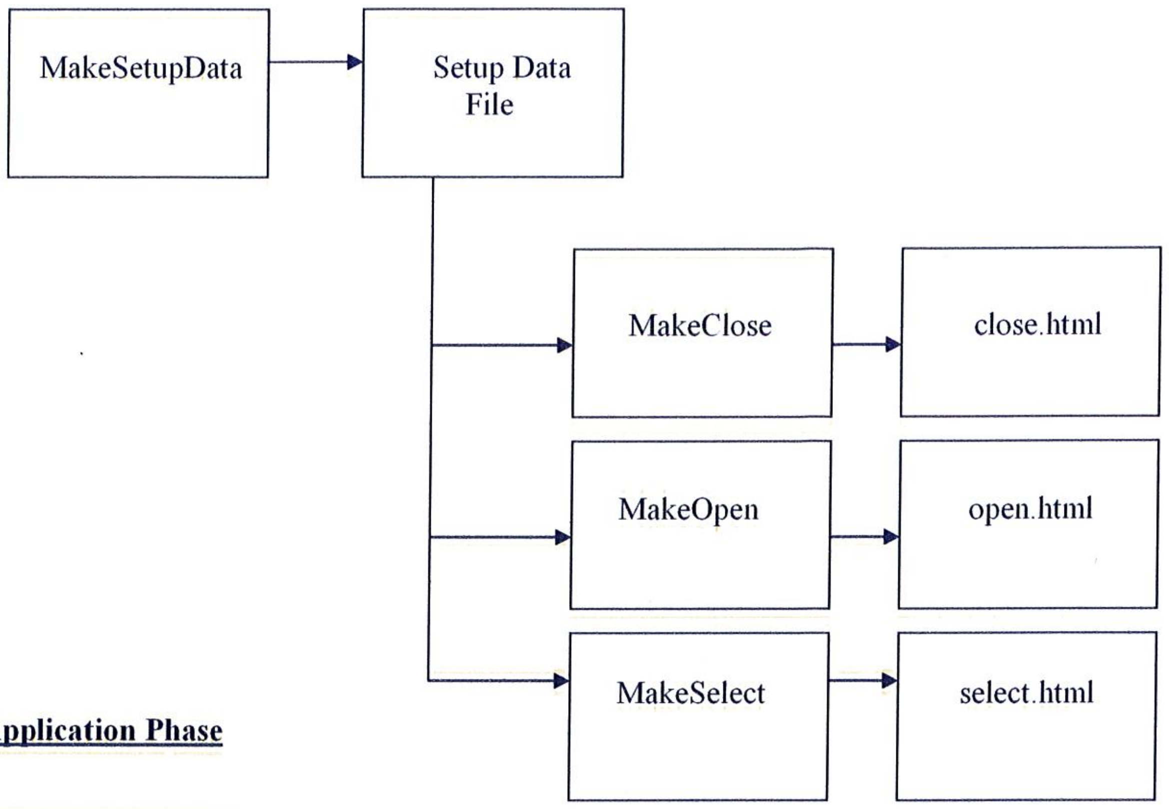
Python, How to program: Introducing XML, By H.M. Deitel, P.J. Deitel, J.P. Liperi, and B.A. Weidermann. Copyright 2002, Prentice Hall, Inc

http://www.w3schools.com, Copyright 1999-2007 by Refsnes Data. Accessed on January 3, 2007.

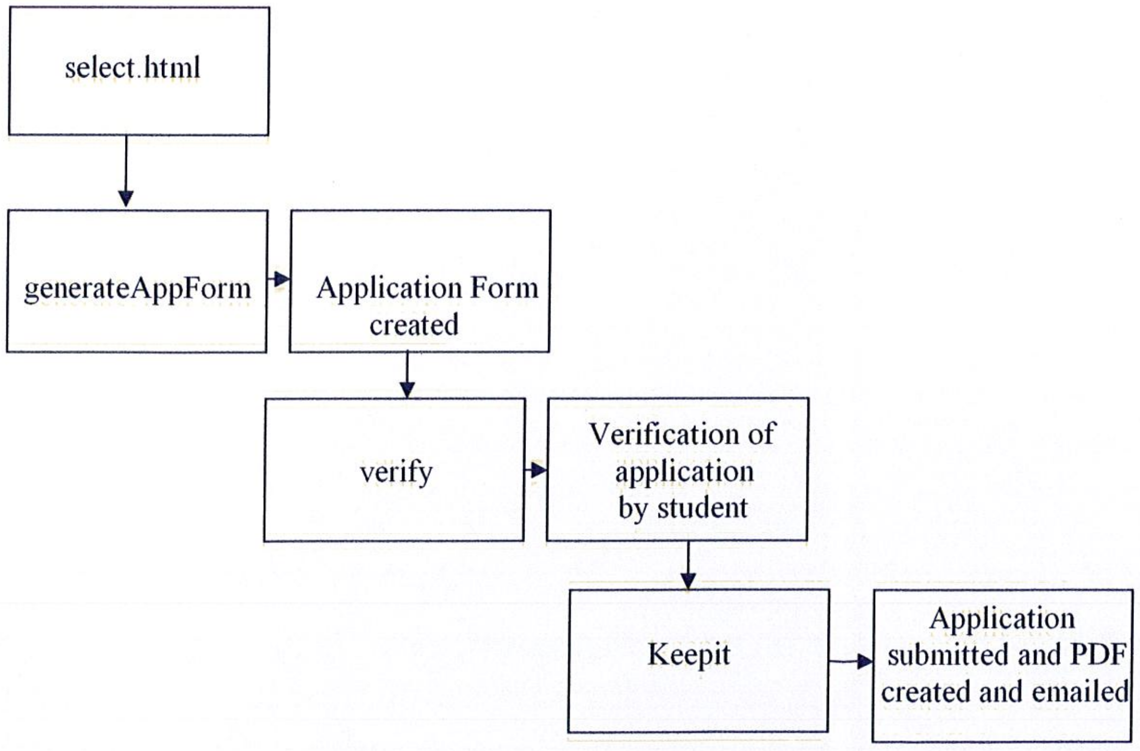
http://www.w3c.com, Copyright 1999-2007 By WWW Communications Ltd. Accessed on February 2006.

Appendix A – contains a basic diagram of the two different phases (setup and application)

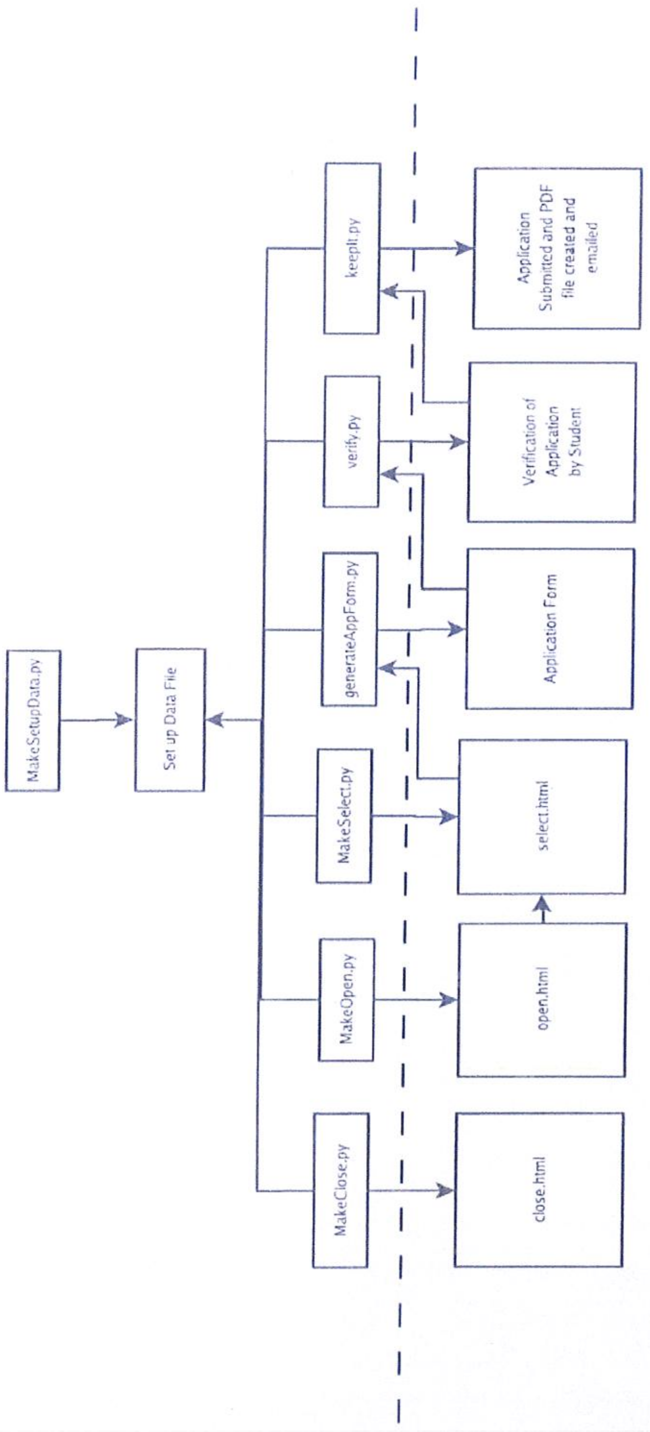
<u>Setup Phase</u>



<u>Application Phase</u>


select.html
generateAppForm
Application Form created
verify
Verification of application by student
Keepit
Application submitted and PDF created and emailed

Appendix B – Contains a brief view of the user's perspective (right side) and the behind the scenes workings of the software (left side) that goes along with the application process.



Appendix C- The following is the software protocol that guides the administrator through the set up phase of the software package.

SSCA application software protocol
This software package was made for the purpose of updating the data that is collected for the SSCA application website. It is broken into two main parts: the setup and application. The software is licensed under the GNU GPL format and is free for use and to be copied as pleased.

Setup Data - The setup data stage is broken into four main pieces. They are as follows: MakeSetupData.py, MakeClose.py, MakeSelect.py, and MakeOpen.py. These parts are all called within a bash shell program called SSCAsetup. The bash script also moves all the required files to the needed locations.

MakeSetupData.py - This is the major part of the setup phase. It gathers the data that is used for the different HTML files and the application phase.

Data file - The first step is entering the year that the data is being collected for. The software then takes that year and checks to see if there is data for that year already in the file; as well as the last year's data. So, if you would enter 2007, the software would check for the 2007 data file. If that doesn't exist it checks to see if there is a data file for 2006. If that doesn't exist then you have to start with a new data file. If either data file is found, then the data for that year is displayed and you are asked if you would like to re-use the data in that file. If you choose to re-use the data then you are given the option to edit the data in the file. If you choose to not re-use the data file a new data file is created. If there isn't a data file found or if you have chose not to re-use the data file, then you have to start by entering the new data for the new data file.

Dates - The first two fields that are entered are the date of the conference and the last day to submit applications. Both fields are verified by the software to make sure that a valid date is entered.

Number of students and faculty sponsors - Then the software asks for the maximum number of students and faculty sponsors. Both fields are a numeric number.

Student Information - The next field that has to be entered is the student information. This contains basic information that is to be collected on the application for the student. The software is setup so that the First Name, Last Name, Hometown, Major and Email should be the first five data fields. If these fields are changed, subsequent changes are needed to be made to the verify.py and Keepit.py files. First you are asked if you want to change anything on the list of items that are normally used. If not, the software continues. If you want to change something then the software will ask if you would like to add or remove an item. When adding to the student information field the user will be asked what they would like to add and where they would like it placed. When removing from the list they are asked which one they would like removed based on the position of the item being removed.

Presentation types - The next field is the presentation types. These are set up as a class object of PresentationTypes.py. They have two attributes: type and description. The software will ask if you want to change the list that is normally used. If not, the software will continue to the next step. If you would like to change the list then you are asked if you would like to add or remove an object from the list. When the user wants to add a new presentation they are asked for the type of presentation, the description of that presentation, and where on the list they would like that presentation to be. When removing a presentation they are asked which one they would like to be removed based on the position of the object being removed.

Equipment Needs - The next field is the equipment needs of the student presenters. This is similar to the student information in that it is given as a list of items. The software will ask if you would like to change the list that is given. If not the software continues. If you would like to change the list then you are given the option to add or remove an item. When adding an item your asked what you would like to add and where you would like it to be on the list. When removing an item you are asked what item you would like to remove based on the position of the item.

Editing Fields - After these fields are entered or if you have decided to use the previous data, you are then given the option to edit any of the fields. Each field is given a number, corresponding to a field that is editable. The list is as follows:

0. Change nothing
1. Conference Dates
2. Students Allowed
3. Sponsors Allowed
4. Student Info
5. Presentation Types
6. Equipment Provided
7. Display information again

When selecting 1-6 you go through the same process as if you were entering the data for the first time. The '7' option displays all of the information that has been entered for the current data file, whether it is a new or old version of the data. The '0' option finishes the creation of the setup data by pickling the data into a data file for use by other files.

Appendix D – contains the source code for the program.

Name of the file and what the file is used for:

AppForm.py	Class structure used to hold setup data.
Date.py	Class structure used to check dates in the setup phase.
GenerateAppForm.py	Creates the application form based off of the setup data.
Helper.py	File used for holding several functions used by most of the other files.
Keepit.py	The final submission page creates PDF and other files used for administrative purposes.
MakeClose.py	Creates the close.html file.
MakeOpen.py	Creates the open.html file.
MakeSelect.py	Creates the select.html file and is the link to the generateAppForm.py for creating the application form.
MakeSetupData.py	Creates the setup data file, SetupData<year>.
PresentType.py	Class structure used in the setup data phase. Allows for presentations to have a name and a description.
SSCAsetup	Bash shell script used for calling the setup programs.
TagConvert.py	Class used for defining special symbols and translates them
TexBase.py	Class used to create the general outline of the PDF file
Verify.py	Used to create the verification form after the generateAppForm.py and also contains the link to Keepit.py

SSCAsetup

```
#!/bin/bash

python MakeSetupData.py
python MakeClose.py
python MakeOpen.py
python MakeSelect.py

for name in *.html ; do
  mv $name ../
done

cp SetupData`date +%Y` ../scripts/setUpData
```

```
#    SSCA application process for Bemidji State University
#    Copyright (C) 2007  Charles Hofer

#    This program is free software; you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation; either version 2 of the License, or
#    (at your option) any later version.

#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.

#    You should have received a copy of the GNU General Public License along
#    with this program; if not, write to the Free Software Foundation, Inc.,
#    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.


import sys
import string
from Date import *
from PresentType import *

class AppFormData:

# Used with declaration with all of the AppForm Data
  def __init__(self, year, begDate, endDate, maxStudent, maxFaculty,
"              studentInfo, presentationTypes, presentationNeeds):
    self.year = year
    self.begDate = self.checkDate(begDate)
    self.endDate = self.checkDate(endDate)
    self.maxStudent = maxStudent
    self.maxFaculty = maxFaculty
    self.studentInfo = studentInfo
    self.presentationTypes = presentationTypes
    self.presentationNeeds = presentationNeeds

# Used for starting an empty AppFormData member
  def __init__(self):
    self.year = 0

# Displays the AppForm Data
  def Display(self):
    print "Current Year: %s\n" %(self.year)
    print "Beginning Date: %s" %(self.begDate)
    print "End Date: %s\n" %(self.endDate)
    print "Maximum number of Students: %s" %(self.maxStudent)
    print "Maximum number of faculty sponcers: %s" %(self.maxFaculty)
    print "List of Student Information: %s\n" %(self.studentInfo)
    print "List of Presentation Types:"
    for item in self.presentationTypes: # Used to print out all the different types
      print item.display()
    print "List of Presentation Needs: %s" %(self.presentationNeeds)
```

```
# Checks the dates
  def checkDate(self, testDate):
    month, date, year = str(testDate.split("/"))
    return Date(month, int(date), int(year))


# Gets and Sets used to return and set the value of the information that is
#  stored in them periodically throughout the code.
  def setYear(self, year):
    self.year = year

  def getYear(self):
    return self.year

  def setBegDate(self, begDate):
    self.begDate = begDate

  def getBegDate(self):
    return self.begDate

  def setEndDate(self, endDate):
    self.endDate = endDate

  def getEndDate(self):
    return self.endDate

  def setMaxStudent(self, maxStudents):
    self.maxStudent = maxStudents

  def getMaxStudent(self):
    return self.maxStudent

  def setMaxFaculty(self, maxFaculty):
    self.maxFaculty = maxFaculty

  def getMaxFaculty(self):
    return self.maxFaculty

  def setStudentInfo(self, studentInfo):
    self.studentInfo = studentInfo

  def getStudentInfo(self):
    return self.studentInfo

  def setPresentationTypes(self, presentationTypes):
    self.presentationTypes = presentationTypes

  def getPresentationTypes(self):
    return self.presentationTypes

  def setPresentationNeeds(self, presentationNeeds):
    self.presentationNeeds = presentationNeeds
```

```
def getPresentationNeeds(self):
  return self.presentationNeeds
```

```
class Date:

  def __init__(self, month, day, year):

    monthNum = checkMonth(month)
    if(monthNum != -1):
      self.month = month
    else:
      raise ValueError, "Invalid value for month: %m" %(month)

    if year > 0:
      self.year = year
    else:
      raise ValueError, "Invalid value for year: %y" %(year)

    self.day = self.checkDay(day, monthNum)

  def __init__(self):
    self.month = "Month"
    self.day = 0

  def display(self):
    print "%d %d, %d" %(self.month, self.day, self.year)

  def checkMonth(self, month):
    monthsInYear = ['0','January','February','March','April','May',
"       'June','July','August','September','October','November','December']
    numMonth = 0
    for item in monthsInYear:
      if (month == item):
        return numMonth
      numMonth += 1
    return -1

  def checkDay(self, testDay, monthNum):
    daysPerMonth = [0,31,28,31,30,31,30,31,31,30,31,30,31]
    daysInMonth = daysPerMonth[monthNum]
    if ( 0 < int(testDay) <= int(daysInMonth)):
      return testDay
    elif monthNum == 2 and testDay == 29 and \
        (self.year % 400 == 0 or
         self.year % 100 != 0 and self.year % 4 == 0):
      return testDay
    else:
      return -1

  def setDay(self, day):
    self.day = day

  def setMonth(self, month):
    self.month = month

  def setYear(self, year):
    self.year = year

  def getDay(self):
    return self.day

  def getMonth(self):
    return self.month

  def getYear(self):
    return self.year
```

```
#!/usr/bin/python


import cgi
import time
import cPickle
import string
from Helper import *
from AppForm import *
from PresentType import *


def buildPage( studentCount, sponsorCount, currentAppForm ):
#  printHeader( "Application Form" )
  year = currentAppForm.getYear()
  confDate = currentAppForm.getBegDate()
  endDate = currentAppForm.getEndDate()
  studentInfo = currentAppForm.getStudentInfo()
  presentationTypes = currentAppForm.getPresentationTypes()
  presentationNeeds = currentAppForm.getPresentationNeeds()
  num = 0
  for item in studentInfo:
    num += 1
  stuIdNum = num * studentCount

  presentationNum = 0
  for item in presentationTypes:
    presentationNum += 1

  presentationNeedNum = 0
  for item in presentationNeeds:
    presentationNeedNum += 1

  sponsorNum = 4 * sponsorCount # The 4 is the number of things that are asked for on the
sponsor

  print """Content-type: text/html

<?xml version  = "1.0" encoding = "UTF-8"?>
<!DOCTYPE html PUBLIC
   "-//W3C//DTD XHTML 1.0 Strict//EN"
   "DTD/xhtml1-strict.dtd">
<html xmlns = "http://www.w3.org/1999/xhtml">
<head><title>Application Form</title>

<script language="javascript">
<!--"""
  print "function checkAll() {"
# Checks if title has a value.
  print '  if(document.getElementById("title").value == "") {'
  print '    alert("You have to enter a Title!");'
  print '  return'
  print '  }'
# Checks if the abstract has a value.
```

```
  print '  if(document.getElementById("abstract").value == "") {'
  print '    alert("You have to enter an Abstract!");'
  print '  return'
  print '  }'
# Checks if the student table is completed.
  print '  var id="stu";'
  print "  for (i=1;i<=" + str(stuIdNum) + ";i++) {"
  print '    id = id.concat(i)'
  print '    if(document.getElementById(id).value == "") {'
  print '      alert("The student information is incomplete!");'
  print '      return'
  print '    }'
  print '    var id = "stu"'
  print '  }'
# Checks to see if there was a radio button selected for presentation types.
  print '  var radioNum = 0'
  print '  var id="presT";'
  print "  for (i=1;i<=" + str(presentationNum) + ";i++) {"
  print '    id = id.concat(i)'
  print '    if(document.getElementById(id).checked == true) {'
  print '      var radioNum = 1'
  print '    }'
  print '    var id = "presT"'
  print '  }'
  print '  if(radioNum == 0) {'
  print '    alert("No presentation type is selected!");'
  print '    return'
  print '  }'
# Checks to see if the sponsor table is completed.
  print '  var id="spon";'
  print "  for (i=1;i<=" + str(sponsorNum) + ";i++) {"
  print '    id = id.concat(i)'
  print '    if(document.getElementById(id).value == "") {'
  print '      alert("The sponsor information is incomplete!");'
  print '      return'
  print '    }'
  print '    var id = "spon"'
  print '  }'
# Everything is complete.
  print '  var message = "  All required fields are completed.  You can now submit your
application."'
  print '  alert(message);'
  print '}'


  print """
// -->
</script>

</head>

<body>
```

```
<img src="http://cs.bemidjistate.edu/hofe1cha/images/Header.gif" width="800" height="144"
alt="Student
  Scholarship and Creative Achievement Conference"/>"""


  print "  <h2>" + str(year)
  print """ Presenter Application Form</h2>
  <p>This year's conference will be held on """
  print str(confDate) + ", " + str(year)
  print """.  We require both an electronic and printed application by
          <span style = "font-weight: bold">"""
  print str(endDate) + ", " +str(year)
  print """</span>.  You will be able to make the electronic submission
  through this website.  Furthermore, you will have the opportunity to
  print your hard-copy once your electronic version has been
  submitted.
  </p>

  <p><span style = "font-weight: bold">Recommendation:</span>
     This form does not support spell checking or grammar checking.
     We recommend that you use a word processor to create your abstract.
     Then ask your advisor to review what you have written.  After
     receiving your advisor's approval, use copy and paste to enter your
     Abstract.
     Please note that once you submit your proposal you WILL NOT be able
     to access it to make corrections. It is
     important that you edit and make a copy of your proposal BEFORE
     you submit it.
     </p>
  <form method = "post" action = "/cgi-bin/thesis/verify.py">
         <p>Title: <input type = "text" id ="title" name = "title" size=50 /> </p>

         <p>Abstract:  Please enter a description of your project
            <span style = "font-weight: bold"> totaling not more than
               150 words</span>
            including your working thesis or hypothesis, methodological
            plan, and a brief statement
            regarding the importance of the project.  The abstract will
            be printed in the conference booklet; be sure it is in good
            form.</p>
            <p>Note that if you need to include any special characters
            or notation in your title or abstract,
            you should <a
            href="http://cs.bemidjistate.edu/hofe1cha/learn.html"
            target="_blank">learn how.</a></p>

         <textarea id ="abstract" name="abstract" Rows=20 Cols=60 wrap=virtual></textarea>
         <p>

         Please enter the following information for
            <span style = "font-weight: bold">each presenter</span>.
         Note that each presenter <span style = "font-weight:
         bold">must</span> supply a """
  count = 0
  for item in studentInfo:
    item = item[0:len(item)-1]
    if count == 0:
      print item
    elif count == len(studentInfo) - 1:
      print " and " + item
    else:
      print ", " + item
    count += 1

  print """for the
         registration process to succeed.
         <p>Also some E-mail addresses are unable to recieve the automated E-mail that is
sent out.
"            It is recomended that BSU E-mail accounts are used.</p>
         <table width = "100%" border = "3" name = "studentTable">
         <tr>
           <th></th>"""

# Used to make fields for entering the student info
  for item in studentInfo:
    item = item[0:len(item)-1]
    print "            <th>%s</th>" %(item)
  print "            <th>Student Type</th>"
  print "          </tr>"

  count = 1 # Used for the student Id's for each of there cells
  for i in range(0,studentCount):
    c = i + 1
    print """        <tr>
         <td>%d</td>""" %(c)
    for item in studentInfo:
      item = item[0:len(item)-1]
      print '          <td><input type = "text" id = "stu%d" name = "%s%d" value = ""></td>'
%(count, item, c)
      count += 1

    print '          <td><select name = "stuType%d">' %(c)
    print '              <option value="Regular Student">Regular Student</option>'
    print '              <option value="Honors Student">Honors Student</option>'
    print '              <option value="Graduate Student">Graduate Student</option>'
    print '          </select></td>'
    print "        </tr>"


  print """</table>
          </p>

         <p> <span style = "font-weight: bold">Please check the
         appropriate button.  <br>Please note that each presentation,
         panel and performance
         will have a twenty minute time slot.</span> <br>"""
# Presentation Types
```

```
  idNum = 1
  for item in presentationTypes:
    print '          <input type=radio id = "presT%d" name="presType" value="%s">' %(idNum,
item.presentType)
    print '          <span style = "font-weight: bold">' + item.presentType + ':</span>'
    print '            ' + item.discription + '  <br>'
    idNum += 1

  print """
         </p>


        <p>
         <span style = "font-weight: bold">Indicate your
         equipment needs.</span><br>"""

# Presentation Needs
  idNum = 1
  for item in presentationNeeds:
    item = item[0:len(item)-1]
    print '          <input type=checkbox id = "presN%d" name=equip value="%s">%s<br>' %
(idNum, item, item)
    idNum += 1

  print """
"        <p></p>
         If you have any special requests, concerns or equipment
         needs not listed above, please enter
         them below:<br>
         <textarea id ="special" name="specialNeeds" rows=6 cols=40 wrap=virtual></textarea>
         </p>
         <p>
         <span style = "font-weight: bold">
         Faculty Sponsor Information:</span>
         <br>
         <table width = "100%" border = "3" name = "facultyTable">
         <tr>
           <th></th>
           <th>First Name</th>
           <th>Last Name</th>
           <th>E-mail</th>
           <th>Phone Number</th>
         </tr> """

  idNum = 1
  sponsorData = ["sponsorFName", "sponsorLName", "sponsoremail", "sponsorphone"]
  for i in range(0,sponsorCount):
    c = i + 1
    print """          <tr>
          <td>%d</td>""" %(c)
    for item in sponsorData:
      print '          <td><input type ="text" id = "spon%d" name ="%s%d" value =""></td>'
%(idNum, item, c)
      idNum += 1
    print'          </tr>'


  print """</table>
           </p>
          <p><span style = "font-weight: bold">
             Please check to see if all required fields are correct
             by pressing the Pre Submission Check button.</span></p>
          <p>
          <input type="button" onclick=checkAll() value="Pre Submission Check" />
          </p>
          <p>
          <input type = "submit" value ="Submit Application for Verification" />
          </p>

    </form>
  </body>
</html>"""



def main():
  if (os.path.exists ("setUpData")):
    currentAppForm = AppFormData()
    fName = open("setUpData", 'r')
    currentAppForm = cPickle.load(fName)


    pairs = cgi.FieldStorage()

    studentCount = eval( pairs[ "countStudents" ].value )
    sponsorCount = eval( pairs[ "countFaculty" ].value )

    buildPage(studentCount, sponsorCount, currentAppForm)

main()
```

```
#    SSCA application process for Bemidji State University
#    Copyright (C) 2007  Charles Hofer

#    This program is free software; you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation; either version 2 of the License, or
#    (at your option) any later version.

#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.

#    You should have received a copy of the GNU General Public License along
#    with this program; if not, write to the Free Software Foundation, Inc.,
#    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

import os
import time
from tagConvert import *

def pdfAnnualHeader():
  return """$7^{th}$ Annual Student Scholarship and Creative Achievement Conference"""

def emailFrom():
  return "Charles <hofe1cha@cs.bemidjistate.edu>"

def studentEmailText():
  studentText = """
Dear Student,

Your application for the Student Scholarship and
Creative Achievement day presentations was successfully
submitted.  The attatchment contains a copy of the pdf
with the information about your presentation.  Thanks
for applying and good luck on the presentation.

Note: You don't have to get the signitures on this copy!
"""
  return studentText

def facultyEmailText():
  facultyText = """
Dear Faculty Member,

You have been noted as a faculty sponsor on an
Application for the Student Scholarship and Creative
Achievement day presentation.  The pdf attatchment has
the information about the presentation.  Please keep a
copy of this for your records.
"""
  return facultyText

def selectDirectory():
   return "http://cs.bemidjistate.edu/hofe1cha/"

 def wwwDirectory():
   return "/home/hofe1cha/www/"

 def scriptDirectory():
   return "/cgi-bin/thesis/"

 def findAppYear():
   year = time.strftime("%Y")
   lastYear = int(year) - 1
   fileName1 = "SetupData" + str(year)
   fileName2 = "SetupData" + str(lastYear)
   if ( os.path.exists (fileName1)): # current year
     return fileName1
   elif ( os.path.exists (fileName2)): # last year
     return fileName2
   else:  # file does not exist
     return 0

 def printHeader ( title ):
   print """Content-type: text/html

   <?xml version  = "1.0" encoding = "UTF-8"?>
   <!DOCTYPE html PUBLIC
      "-//W3C//DTD XHTML 1.0 Strict//EN"
      "DTD/xhtml1-strict.dtd">
   <html xmlns = "http://www.w3.org/1999/xhtml">
   <head><title>%s</title></head>

   <body>

   <img src="http://cs.bemidjistate.edu/hofe1cha/images/Header.gif" width="800" height="144"
 alt="Student
   Scholarship and Creative Achievement Conference"/>""" % title

 def printHeaderFile(fName):
 #    fName = open(fileName,'w')
     fName.write('<?xml version  = "1.0" encoding = "UTF-8"?>\n')
     fName.write('<!DOCTYPE html PUBLIC\n')
     fName.write('    "-//W3C//DTD XHTML 1.0 Strict//EN"\n')
     fName.write('    "DTD/xhtml1-strict.dtd">\n')
     fName.write('<html xmlns = "http://www.w3.org/1999/xhtml">\n')
     fName.write('<head><title>\n')
     fName.write('Student Scholarship and Creative Achievement Application\n')
     fName.write('</title></head>\n\n')

     fName.write('<body>\n')
     fName.write('<img src="images/Header.gif" width="800" height="144" alt="Student\n')
     fName.write('Scholarship and Creative Achievement Conference">\n\n')

 def cleanTableEntries( theString ):
   """This functions escapes all of tex's special characters that are
```

```
    used in tables."""

  theString = theString.replace("\\", "\backslash")
  #  rTable = { ord("{"): "\{", ord("}"): "\}", ord("$"): "\$", ord("#"): "\#", ord("%"): "\
%", ord("&"): "\&", ord("_"): "\_", ord("~"): "\~{\ }", ord("^"): "\^{\ }"}

  theString = theString.replace("{", "\{")
  theString = theString.replace("}", "\}")
  theString = theString.replace("$", "\$")
  theString = theString.replace("#", "\#")
  theString = theString.replace("%", "\%")
  theString = theString.replace("&", "\&")
  theString = theString.replace("_", "\_")
  theString = theString.replace("~", "\~{\ }")
  theString = theString.replace("^", "\^{\ }")

  return theString

def markup2TeX( theString ):
  converter = TagConvert()
  retString = ""
  tagStart = 0
  tagEnd = 0
  mathOpenCount = 0 #keeps track of the number of tags have math mode
                    #open

  # all special chars in tex need to be escaped to work out
  theString = theString.replace("\\", "\backslash")
  theString = theString.replace("{", "\{")
  theString = theString.replace("}", "\}")
  theString = theString.replace("$", "\$")
  theString = theString.replace("#", "\#")
  theString = theString.replace("%", "\%")
  ##theString = theString.replace("&", "\&")
  theString = theString.replace("_", "\_")
  theString = theString.replace("~", "\~{\ }")
  theString = theString.replace("^", "\^{\ }")

  ## The following is probably better as a dictionary or class, but two
  ## parallel arrays will have to do for now
  startTagList = [ "<", "&" ]
  endTagList =   [ ">", ";" ]

  while  True :
    #find the next start tag
    firstAngle = theString.find ("<", tagStart)
    firstAmp = theString.find ("&", tagStart)
    tagStart = firstAmp
    if tagStart == -1 :
      tagStart = firstAngle
    elif firstAngle != -1 :
      tagStart = min( tagStart, firstAngle )
    #if there isn't any, we process to the end of the string and we are done
    if tagStart == -1 :
      retString += theString[tagEnd:]
      break;

    #now add from the previous end tag to the next start tag
    retString += theString[tagEnd:tagStart]

    #Determine the tag's position in the parallel arrays
    tagListIndex = startTagList.index( theString[ tagStart ] )


    tagEnd =  theString.find(endTagList[ tagListIndex ], tagStart)
    tag = theString[tagStart:tagEnd+1]


    if ( theString[ tagStart ] == "<" ) :
      replacement = converter.convert(tag.lower(),"tex")
      if tag.lower() == replacement :  ## can't convert, assume lone <
        retString += "$<$"
        tagStart = tagStart + 1
"       tagEnd = tagStart
"       continue
    elif ( theString[ tagStart ] == "&" ):
      replacement = converter.convert(tag,"tex")
      if tag == replacement :  ## can't convert, assume lone &
        retString += "$\&$"
        tagStart = tagStart + 1
"       tagEnd = tagStart
"       continue

    if mathOpenCount == 0 or replacement.find("$") == -1 :
        # math not open or this token doesn't involve math mode
      retString += replacement
    elif mathOpenCount >= 1 and replacement[0] == "$"  \
                            and replacement[-1] == "$" :
        # math mode is open and this token is a special character
      retString += replacement[1:-1]
    elif mathOpenCount == 1 and replacement.find("$") != -1  \
                            and tag.find("/") != -1:
        # math open and this token  closes math mode
      retString += replacement
    elif mathOpenCount >= 1 and replacement.find("$") != -1  \
                            and tag.find("/") == -1:
"       # math mode open and this token starts a new math mode mark
      retString += replacement[1:]
    elif mathOpenCount >= 1 and replacement.find("$") != -1  \
                            and tag.find("/") != -1:
"       # math mode open and this token ends a math mode mark
      retString += replacement[:-1]

    if replacement[0] == "$" and replacement[-1] != "$":    # opened one up
      mathOpenCount += 1
    elif replacement[0] != "$" and replacement[-1] == "$":
      mathOpenCount -= 1            # closed one off

    tagStart = tagEnd + 1
    tagEnd = tagStart


  retString = retString.replace(">", "$>$")

  return retString
```

```
#!/usr/bin/python

#    SSCA application process for Bemidji State University
#    Copyright (C) 2007  Charles Hofer

#    This program is free software; you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation; either version 2 of the License, or
#    (at your option) any later version.

#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.

#    You should have received a copy of the GNU General Public License along
#    with this program; if not, write to the Free Software Foundation, Inc.,
#    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

import cgi
import os
import string
import cPickle
from Helper import *
from texBase import *
from AppForm import *
from PresentType import *
# Imports used for the email
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email.Utils import COMMASPACE, formatdate
from email import Encoders


# Makes the CVS file
def saveCVS( pairs, fileBase, currentAppForm ):
  maxStudent = currentAppForm.getMaxStudent()
  maxFaculty = currentAppForm.getMaxFaculty()
  presentationTypes = currentAppForm.getPresentationTypes()
  equipNeedList = currentAppForm.getPresentationNeeds()
  stuList = currentAppForm.getStudentInfo()
  stuInfo = []
  for item in stuList:
    stuInfo.append(item[0:len(item)-1])
  equipmentNeed = []
  for item in equipNeedList:
    equipmentNeed.append(item[0:len(item)-1])


  CVSString = ""

  CVSString+='"'+  pairs["title"]  + '",'
```

```
CVSString+='"'+ pairs["abstract"] + '",'

for i in range (int(maxStudent)):
  if pairs.has_key(stuInfo[0] + str(i+1)):
    for item in stuInfo:
      CVSString+='"'+ pairs[item + str(i+1)] + '",'
    CVSString+='"'+ pairs["stuType" + str(i+1)] + '",'
  else:
    for item in stuInfo:
      CVSString+='"",'
    CVSString+='"",'

contribType = pairs [ "presType"]
for item in presentationTypes:
  if contribType == item.presentType :
    CVSString += '1,'
  else:
    CVSString += '0,'

if pairs.has_key("equip"):
  theList = pairs[ "equip" ]
  count = 0
  equipList = []
  for item in equipmentNeed:
    equipList.append(0)
    count +=1
  for item in theList :
    for i in range(0, count):
      if item == equipmentNeed[i]:
        equipList[i] = 1
  for item in equipList:
    CVSString += str(item) +","

else:
  for item in equipmentNeed:
    CVSString += '0,'

if pairs.has_key("specialNeeds"):
  CVSString +=  '"' + pairs [ "specialNeeds" ] + '",'
else:
  CVSString += '"",'

for i in range (int(maxFaculty)):
  c = str(i+1)
  if pairs.has_key("sponsorFName" + c):
    CVSString+='"'+ pairs["sponsorFName" + c] + '",'
    CVSString+='"'+ pairs["sponsorLName" + c] + '",'
    CVSString+='"'+ pairs["sponsoremail" + c] + '",'
    CVSString+='"'+ pairs["sponsorphone" + c] + '",'
  else:
    CVSString+='"","","","",'
  theFile = open ( wwwDirectory() + "CSV/" + fileBase, "w")
  theFile.write(CVSString)
  theFile.close()

def writeStudentContact(pairs, file, studentNumber, stuInfo):
  numString = str(studentNumber)
  if pairs.has_key(stuInfo[0] + numString ):
    file.write("\\item[ ")
    file.write(cleanTableEntries(pairs[stuInfo[0] + numString ]) + ' ')
    file.write(cleanTableEntries(pairs[stuInfo[1] + numString ]) + ' ] ')
    for i in range(4, len(stuInfo), 1):
      if (i == stuInfo):
        file.write(cleanTableEntries(pairs[stuInfo[i] + numString ]) + '\n')
      else:
        file.write(cleanTableEntries(pairs[stuInfo[i] + numString ]) + ', ')


def writeStudentTableStuff(pairs, file, studentNumber, stuInfo):
  numString = str(studentNumber)
  if pairs.has_key(stuInfo[0] + numString ):
#    file.write("\\item[ ")
    file.write(cleanTableEntries(pairs[stuInfo[0] + numString ]) + ' ')
    file.write(cleanTableEntries(pairs[stuInfo[1] + numString ]) + ' & ')
    file.write(cleanTableEntries(pairs[stuInfo[2] + numString ]) +' & ')
    file.write(cleanTableEntries(pairs[stuInfo[3] + numString ]) +' & ')
    file.write(cleanTableEntries(pairs["stuType" + numString ]) +' \\\\ \hline ')

def writeSponsorInfo(pairs, file, sponsorNumber):
  numString = str(sponsorNumber)
  if pairs.has_key("sponsorFName" + numString ):
    file.write(pairs["sponsorFName" + numString ] +' ')
    file.write(pairs["sponsorLName" + numString ] +' & ')
    file.write(cleanTableEntries(pairs["sponsoremail" + numString ]) +' & ')
    file.write(cleanTableEntries(pairs["sponsorphone" + numString ]) +' \\\\ \hline ')

def writeSponsorSignature(pairs, file, sponsorNumber):
  numString = str(sponsorNumber)
  if pairs.has_key("sponsorFName" + numString ):
    file.write ( "\n\\noindent\nI, " + pairs["sponsorFName" + numString] + " " +
pairs["sponsorLName" + numString] + """, agree that this student work has
significant
scholarly merit and is appropriate for presentation at this conference.
I also agree to \\textbf{preview and approve the presentation before}
the conference.""")

    file.write("""\n\n\\vspace{1cm}\n\n\\noindent
Faculty Sponsor Signature\\hrulefill
Date\\makebox[4cm]{\\hrulefill}\n""")
  file.write ( "\n\\vspace{\\lineskip}")

# Makes tex file
def writeTeXFile(pairs, fileBase, maxStudent, maxFaculty, stuInfo):
  tb = texBase()
  file = open ( wwwDirectory() + "texFiles/" + fileBase + ".tex", "w")


  file.write( tb.preamble )

  file.write( "\\markright{ Tracking Number: " + fileBase + " \\hfill }\n" )

  file.write( """\\begin{center}
  {\\Large Application Form for a """ + pairs["presType"] +' } \\\\')

  file.write ( tb.header  )
  file.write ( "\n\\end{center}")
  file.write ( "\n\n \\section*{Student and Presentation Information}")
  file.write ( "\n\n \\noindent\nTitle:  " + markup2TeX(pairs["title"]) +'\n')
  file.write ( "\\vspace{\\lineskip}")

  file.write ( "\n\n \\noindent\nAbstract:  " + markup2TeX(pairs["abstract"]) +'\n')

  file.write ( "\\vspace{\\lineskip}")

  file.write ( "\n\n \\noindent\nPresenters:\\\\" +
               "\\noindent\n" +
"              "\\begin{tabular}{|l|l|l|l|}\\hline\n" +
               "Name & Hometown & Majors & StudentType \\\\ \hline\n")

  for i in range(1, int(maxStudent)):
    writeStudentTableStuff( pairs, file, i, stuInfo )

  file.write ( "\\end{tabular}")

  file.write ( "\\vspace{\\lineskip}")
  file.write ( "\n\n \\noindent\nContact Information:\n" +
"              "\\begin{description}\n" )

  for i in range(1, int(maxStudent)):
    writeStudentContact( pairs, file, i, stuInfo )
  file.write ( "\\end{description}")
  file.write ( "\\vspace{\\lineskip}")

  if pairs.has_key("equip"):

    size = len(pairs["equip"])
    if size > 0:
      file.write("\\noindent\nWe have the following equipment needs:  ")
      file.write(pairs["equip"][0])
      for i in range(1, size):
        file.write(", " + pairs["equip"][i] )
    else:
      file.write("\\noindent\nWe have no equipment needs.\n")


  if pairs.has_key("specialNeeds"):
    file.write ( "\n\n\\vspace{\\lineskip}")
    file.write("\\noindent\nWe have the following special requests:  ")
    file.write(cleanTableEntries(pairs["specialNeeds"]))
  file.write('\\newpage')

  file.write ( "\n\n \\section*{Sponsor Information and Signatures}")
  file.write ( "\n\n \\noindent\nSponsor Information:\\\\" +
               "\\noindent\n" +
               "\\begin{tabular}{|l|l|l|}\\hline\n" +
               "Name & Email & Phone Number \\\\ \hline\n")

  for i in range(1, int(maxFaculty)):
    writeSponsorInfo( pairs, file, i )

  file.write ( "\\end{tabular}")

  file.write ( "\\vspace{\\lineskip}")

  file.write ( "\n\n\\vspace{\\lineskip}")
  for i in range(1, int(maxFaculty)):
    writeSponsorSignature( pairs, file, i )

  file.write (tb.post)
  file.close()

# Makes the PDF file
def makePDF(baseName):
  command =  "cd " + wwwDirectory() + "texFiles/ ; "  \
           + "/usr/bin/pdflatex " + baseName  \
           + ".tex > " + wwwDirectory() + "errors/"+ baseName + ".errors"
  os.system( command )
  os.system( "/bin/mv -f " + wwwDirectory() + "texFiles/" + baseName + ".pdf " +
wwwDirectory() + "pdf/")
  os.system( "/bin/rm -f " + wwwDirectory() + "texFiles/" + baseName + ".aux")
  os.system( "/bin/rm -f " + wwwDirectory() + "texFiles/" + baseName + ".log")

# Grabs the student email addresses
def checkForStudentEmail(pairs, stuInfo, studentNumber, fileName):
  text = studentEmailText()
  subject = "SSCA presentation Aplication"
  for i in range(int(studentNumber)):
    subject = "SSCA presentation Aplication"
    if pairs.has_key(stuInfo[4] + str(i+1)):
      stuAddress = pairs[stuInfo[4] + str(i+1)]
      sendMailPDF(stuAddress, subject, text, fileName)

# Grabs the faculty email addresses
def checkForFacultyEmail(pairs, facultyNumber, fileName):
  text = facultyEmailText()
  subject = "SSCA presentation Aplication"
  for i in range(int(facultyNumber)):
    subject = "SSCA presentation Aplication"
    if pairs.has_key("sponsoremail" + str(i+1)):
      facAddress = pairs["sponsoremail" + str(i+1)]
      sendMailPDF(facAddress, subject, text, fileName)

# Emails the PDF to student and Faculty
def sendMailPDF(to, subject, text, pdfFile, server="localhost"):
  fro = emailFrom() #gets the from address from Helper.py

  msg = MIMEMultipart()
  msg['From'] = fro
  msg['To'] = (to)
  msg['Date'] = formatdate(localtime=True)
  msg['Subject'] = subject

  msg.attach( MIMEText(text) )

  part = MIMEBase('application', "octet-stream")
  part.set_payload( open(pdfFile,"rb").read() )
  Encoders.encode_base64(part)
  part.add_header('Content-Disposition', 'attachment; filename="%s"'
                 % os.path.basename(pdfFile))
  msg.attach(part)

  smtp = smtplib.SMTP(server)
  smtp.sendmail(fro, to, msg.as_string() )
  smtp.close()

def main():
  if (os.path.exists ("setUpData")):
    currentAppForm = AppFormData()
    fName = open("setUpData", 'r')
    currentAppForm = cPickle.load(fName)

  maxStudent = currentAppForm.getMaxStudent()
  maxFaculty = currentAppForm.getMaxFaculty()
  stuList = currentAppForm.getStudentInfo()
  stuInfo = []
  for item in stuList:
    stuInfo.append(item[0:len(item)-1])

  directory = wwwDirectory()
  currentPairs = cgi.FieldStorage()

  fileName = currentPairs [ "trackingNumber" ].value
  fileBase = fileName[:30]
  fileName = directory + "proposals/" + fileBase
  try:
    webFile = open ( fileName, "r")
    pairs = cPickle.load( webFile )
    webFile.close()
  except IOError:
    printHeader("Submission Error")
    print """<p>We do not have any record of a pending submission for you.
       You will need to <a href="%sselect.html">
       start the submission process again</a>.</p>""" %(directory)
    print "</body></html>"
    sys.exit()

  writeTeXFile(pairs, fileBase, maxStudent, maxFaculty, stuInfo)
  makePDF(fileBase)
  saveCVS(pairs, fileBase, currentAppForm)

  attatchmentDir = directory + "pdf/" + fileBase + ".pdf"
  checkForStudentEmail(pairs, stuInfo, maxStudent, attatchmentDir)
  checkForFacultyEmail(pairs, maxFaculty, attatchmentDir)

  printHeader( "Submission Successful." )
  print """<p>Your electronic application has been successfully
           submitted.</p>"""
  print """<p>Your tracking number is %s.  Please use this number in any
           questions that you have about your application.  Note that
"          this number appears on your printable application.</p>""" % fileBase
  print """<p>To complete the submission process you must
           <a href="%spdf/%s.pdf">click and print</a> the
"          application.  Have your faculty sponsor, the department chair
"          and appropriate dean sign the form.  A copy of the form has been
           sent to all students and faculty that are named on the application form
           via email.</p>""" % ("/hofe1cha/", fileBase)
  print "</body></html>"

main()
```

```
from PresentType import *
from AppForm import *
from Helper import *
import sys, cPickle
import string
import time

# Makes close.html file
def main():
  infoF = findAppYear()
  if infoF == 0:
    print "The WebData file could not be found, Please run Make.py"
  else:
    fName = open(infoF, 'r')
    currentAppForm = cPickle.load(fName)
    year = currentAppForm.getYear()
    confDate = currentAppForm.getBegDate()

    fName = open('close.html','w')
    printHeaderFile(fName)

    fName.write('  <h2>' + str(year) + ' Application Information</h2>\n')
    fName.write('  <p>The application process for the ' + str(year) + ' Student Scholarship 
and\n')
    fName.write('  Creative Achievement Conference is now closed.\n')
    fName.write('  We thank for your interest in the conference and look forward to your\n')
    fName.write('  attendance on '+ str(confDate) + ', ' + str(year) + '.\n')
    fName.write('  </p>\n\n')


    fName.write('  <p>\n')
    fName.write(' Return to the \n')
    fName.write('<!--  either -->\n')
    fName.write('  <a href="http://www.bemidjistate.edu/scholar">Student Scholarship and\n')
    fName.write('  Creative Achievement Conference</a> website.\n')
    fName.write('  </p>\n\n')

    fName.write('<a href="http://www.bemidjistate.edu" target="_blank"><img\n')
    fName.write('src="images/bemidji_logo.gif" alt="" width="216" height="63"\n')
    fName.write('border="0"></a>\n\n')

    fName.write('</body></html>\n')

main()
```

```
from PresentType import *
from AppForm import *
from Helper import *
import sys, cPickle
import string
import time

# Makes open.html file
def main():
  infoF = findAppYear()
  if infoF == 0:
    print "The WebData file could not be found, Please run Make.py"
  else:
    fName = open(infoF, 'r')
    currentAppForm = cPickle.load(fName)
    year = currentAppForm.getYear()
    confDate = currentAppForm.getBegDate()
    endDate = currentAppForm.getEndDate()


    fName = open('open.html','w')
    printHeaderFile(fName)

    fName.write('  <h2>')
    fName.write(str(year))
    fName.write(' Student Scholarship and Creative Achievement Conference\n')
    fName.write('     Application Site</h2>\n\n')

    fName.write("  <p>This year's conference will be held on ")
    fName.write(str(confDate) + ', ' + str(year) + '.  We\n')
    fName.write('  require both an electronic and printed application by \n')
    fName.write('            <span style = "font-weight: bold">\n')
    fName.write(str(endDate) + ', ' + str(year) + '</span>.  ')
    fName.write('You will be able to make the electronic submission\n')
    fName.write('  through this website.  Furthermore, you will have the opportunity to\n')
    fName.write('  print your hard-copy once your electronic version has been\n')
    fName.write('  submitted.\n')
    fName.write('  </p>\n\n\n')


    fName.write("""<p><span style = "font-weight: bold">A Head's Up:</span>\n""")
    fName.write('    The application \n')
    fName.write('    process does not support spell checking or grammar checking.\n')
    fName.write('    Thus, prior to starting the application process,\n')
    fName.write('    we recommend that you use a word processor to create your abstract.
\n')
    fName.write('    Then ask your advisor to review what you have written.  After\n')
    fName.write("    receiving your advisor's approval, follow the link below and\n")
    fName.write('    use copy and paste to enter your\n')
    fName.write('    abstract.\n')
    fName.write('   <span style = "font-weight: bold">\n')
    fName.write('    Please note that once you submit your proposal you WILL NOT be able
\n')
    fName.write('    to access it to make corrections (although you can submit another\n')
    fName.write('     one). \n')
    fName.write('     It is\n')
    fName.write('     important that you edit and make a copy of your proposal BEFORE\n')
    fName.write('     you submit it.\n')
    fName.write('  </span>\n')
    fName.write('     </p>\n\n')

    fName.write('  <p>\n')
    fName.write('  You can \n')

    fName.write('<!--  either -->\n')
    fName.write('  <a href="select.html">create a new application</a>.<br>\n')
    fName.write('  <!--\n')
    fName.write('  <a href="modify.html">an existing application</a> if you know its\n')
    fName.write('  tracking number.<br> -->\n')
    fName.write('  or<br>\n')
    fName.write('   Return to the\n')
    fName.write('   <!--  either -->\n')
    fName.write('     <a href="http://www.bemidjistate.edu/scholar">Student Scholarship\n')
    fName.write('     and Creative Achievement Conference</a> website.\n')
    fName.write('         </p>\n\n')

    fName.write('"      <a href="http://www.bemidjistate.edu" target="_blank"><img\n')
    fName.write('"      src="images/bemidji_logo.gif" alt="" width="216" height="63"\n')
    fName.write('"      border="0"></a>\n\n')


    fName.write('</body></html>')


    fName.close()

main()
```

```
from PresentType import *
from AppForm import *
from Helper import *
import sys, cPickle
import string
import time

def main():
  infoF = findAppYear()
  if infoF == 0:
    print "The WebData file could not be found, Please run Make.py"
  else:
    fName = open(infoF, 'r')
    currentAppForm = cPickle.load(fName)
    maxStudent = currentAppForm.getMaxStudent()
    maxFaculty = currentAppForm.getMaxFaculty()

    fName = open('select.html','w')
    printHeaderFile(fName)

    fName.write('  <h2>Pre-application Information</h2>\n\n')


    fName.write('<form method = "post" action = "/cgi-bin/thesis/generateAppForm.py">\n\n')

    fName.write('<p>How many student presenters are there for this contribution?\n\n')

    count = 1
    fName.write('<select name = "countStudents"  >\n')
    for num in range(int(maxStudent)):
      fName.write('            <option value = ' + str(count))
      if(count == 1):
        fName.write(' Selected >1\n')
      else:
        fName.write(' >' + str(count) + '\n')
      count += 1
    fName.write('          </select>\n')
    fName.write(' </p>\n')

    count = 1
    fName.write('<p>How many faculty advisors are there for your work?  \n')
    fName.write('<select name = "countFaculty"  >\n')
    for num in range(int(maxFaculty)):
      fName.write('            <option value = ' + str(count))
      if(count == 1):
        fName.write(' Selected >1\n')
      else:
        fName.write(' >' + str(count) + '\n')
      count += 1
    fName.write('          </select>\n\n')

    fName.write('</p>\n\n')

    fName.write('<input type = "submit" value ="Continue with the Application" >\n\n')
    fName.write('</form>\n\n')

    fName.write('</body></html>\n\n')

main()
```

```
# This program is set up to make the text file for the input of data into the
#  Web page.
#
# Data is entered into the file WebData + conference year. EX = WebData2007.
#
# Each of the different parts of the file seperated by a comment that has the : before it
#   and after it as a delimiter for the text file.  This is used so that the actually
program
#   to write the Web site knows not to read in these lines and will proceed to the next
part
#   of the web design.
#
# Created By: Charles A. Hofer
# Last Edited: February 21, 2007
# Last Edited by: Charles A. Hofer

import sys, cPickle
import string
import os        # OS library is used to check to see if the file exists.
from AppForm import *
from PresentType import *
from Date import *

# Used to grab the information from the files that provide the information
def GrabFiles(fileName):
  fileName2 = './' + fileName
  if(os.path.exists(fileName2)):
    useFile = open(fileName, 'r')
  else:
    dummyFile = open (fileName, "w")
    dummyFile.close()
    useFile = open(fileName, 'r')
  useList = []
  num = 1
  for line in useFile:
    print str(num) + ". " + line
    useList.append(line)
    num += 1
  lChange = raw_input ("Would you like to change this list (y or n)? ")
  while ( lChange == 'Y' or lChange == 'y'):
    FileChange = open (fileName, "w")
    removeAdd = raw_input ("Would you like to Add(a) or Remove(r) an item? ")
    if (removeAdd == 'a' or removeAdd == 'A'):
      add = raw_input ("What would you like added to the list? ")
      add = add + "\n"
      Pos = input ("Where would you like to add it to the list? ")
      useList.insert(Pos-1, add)
    elif (removeAdd == 'r' or removeAdd  == 'R'):
      remove = input ("What item would you like removed(1-" + str(num-1) + ")? ")
      useList.pop(remove-1)
    num = 1
    for i in range(len(useList)):
      FileChange.write ("%1s" %(useList[i]))
      print str(num) + ". " + str(useList[i])
      num += 1
    lChange = raw_input ("Would you like to change this list (y or n)? ")
  return useList

# Changing Presentation Type list and discriptions
def ChangePType(fileName):
  fName = './' + fileName
  num = 1  # keeps track of how many different presentation types there are
  if(os.path.exists(fName)):
    file=open(fName, "r")
    types = cPickle.load(file)
    for item in types:
      print str(num) + ". " + item.presentType + ": " + item.discription
      num +=1
  else:
    print "No previous file for presentation Types"
    types = []
  change = raw_input("Would you like to change this list(y or n)? ")
  while (change == 'Y' or change == 'y'):
    rA = raw_input("Would you like to remove(r) or add(a) an item? ")
    if (rA == 'a' or rA =='A'):
      add = raw_input("What type of presentation would you like to add? ")
      addD = raw_input("What is the discription of the presentation that you would like to
add? ")
      spot = input("Where would you like to put the item in the list? ")
      if (num == 0):
        spot = 0
      nType = PresentType(add, addD)
      types.insert(spot - 1, nType)
    elif (rA == 'r' or rA == 'R'):
      if (num == 0):
        print "No items in List to Remove"
      else:
        remove = input("What item would you like to remove? ")
        types.pop(remove - 1)
    num = 1
    for item in types:
      print str(num) + ". " + item.presentType + ": " + item.discription
      num +=1
    change = raw_input("Would you like to change this list(y or n)? ")
  file = open(fileName, "w")
  cPickle.dump(types, file)
  return types

# Gets the dates for the conference
def Dates(confYear):
  useDate1 = Date()
  noGood = 1
  while (noGood == 1):
    confDay = raw_input ("What is the Date of the conference? ")
    month, day = confDay.split()
    useDate1.setMonth(month)
    useDate1.setDay(day)
    useDate1.setYear(confYear)
    numMonth = useDate1.checkMonth(month)
    if numMonth > 0:
      numDate = useDate1.checkDay(day, numMonth)
      if numDate > 0:
        noGood = 0
      else:
        print "The entered date is not in " + month +"!"
    else:
      print """    The entered Month is not a month, check your spelling
        and months are capitalized or Abbreviated!"""

  useDate2 = Date()
  noGood = 1
  while (noGood == 1):
    endDay = raw_input ("What is the last Day to submit an Application? ")
    month, day = confDay.split()
    useDate2.setMonth(month)
    useDate2.setDay(day)
    useDate2.setYear(confYear)
    numMonth = useDate2.checkMonth(month)
    if numMonth > 0:
      numDay = useDate2.checkDay(day, numMonth)
      if numDay  > 0:
        noGood = 0
      else:
        print "The entered date is not in " + month +"!"
    else:
      print """    The entered Month is not a month, check your spelling
        and months are capitalized or Abbreviated!"""


  return confDay, endDay

# Used to change the max number of students and faculty sponcers
def ChangePeople(personType):
  if personType == 's':
    numPeople = raw_input ("What is the maximum number of students? ")
  else:
    numPeople = raw_input ("what is the maximum number of Faculty Sponcers? ")
  numPeople = numPeople + "\n"
  return numPeople

# Used to check whether the user would like to change the old or current file
def CheckOld(confTitle, currentAppForm):
  print "This is the information that was used in " + confTitle + ":"
  currentAppForm.Display()
  reUse = raw_input ("Would you like to use this information for the project? ")
  return reUse

# Lists the different sections that can be changed by the user if the
#  old data is kept
```

```
def FileChange():
    print "These are the sections that are included in the web data."
    print "NOTE: You are unable to change the conference Year!"
    print "  0. Change Nothing\n"
    print "  1. Conference Dates\n"
    print "  2. Students Allowed\n"
    print "  3. Sponsers Allowed\n"
    print "  4. Student Info\n"
    print "  5. Presentation Types\n"
    print "  6. Equipment Provided\n"
    print "  7. Display information again\n"
    itemChange = input ("What section would you like to change: ")
    return itemChange

# Allows for the change in the document if the file is being reused or if it is being
created
def Changing(currentAppForm, confYear):
  itemChange = FileChange()
  while(itemChange != 0):
    if(itemChange == 1):
      confDay, endDay = Dates(confYear)
      currentAppForm.setBegDate(confDay)
      currentAppForm.setEndDate(endDay)
    elif(itemChange == 2):
      maxStudets = ChangePeople('s')
      currentAppForm.setMaxStudent(maxStudets)
    elif(itemChange == 3):
      maxFaculty = ChangePeople('f')
      currentAppForm.setMaxFaculty(maxFaculty)
    elif(itemChange == 4):
      print """This is the list of things that are needed
        for each of the Students on the Application.\n"""
      stuList = GrabFiles('StudentInfo')
      currentAppForm.setStudentInfo(stuList)
    elif(itemChange == 5):
      print """This is the list of the different forms of
        Presentations that the students can select from.\n"""
      pTypeList = ChangePType('PresentationType')
      currentAppForm.setPresentationTypes(pTypeList)
    elif(itemChange == 6):
      print """This is a list of all the equipment that can
        be provided for the students\n"""
      needList = GrabFiles('EquipmentNeeded')
      currentAppForm.setPresentationNeeds(needList)
    elif(itemChange == 7):
      print "This is the information that is being used for the conference:"
      currentAppForm.Display()
    else:
      print "Not an applicable selection"
    itemChange = FileChange()
```

```
# Makes the file for the specific year
def main():
  confYear = input ("What is the year that the text will be Made for? ")
  confTitle1 = "SetupData" + str(confYear)
  confTitle2 = "SetupData" + str(confYear - 1)
# checks to see if there exists a file for the last two years
# if Yes asks them if they would like to re-use it or not
# if No, the user is then prompted to enter the informaiton in
  if ( os.path.exists (confTitle1)):
    oldFile = open(confTitle1, "r")
    currentAppForm = cPickle.load(oldFile)
    reUse = CheckOld(confTitle1, currentAppForm)
  elif ( os.path.exists (confTitle2)):
    oldFile = open(confTitle2, "r")
    currentAppForm = cPickle.load(oldFile)
    reUse = CheckOld(confTitle2, currentAppForm)
  else:
    currentAppForm = AppFormData()
    reUse = 'N'

# If the reUse is a Yes
  if(reUse == 'Y' or reUse == 'y'):
    currentAppForm.setYear(confYear)


  else:
# concatinates the month, day, and year together
    confDay, endDay = Dates(confYear)
    currentAppForm.setBegDate(confDay)
    currentAppForm.setEndDate(endDay)
# Maximum number of students that are allowed per form
    numOfStu = ChangePeople('s')
    currentAppForm.setMaxStudent(numOfStu)
# Maximum number of faculty sponsers that are allowed per form
    numOfSpo = ChangePeople('f')
    currentAppForm.setMaxFaculty(numOfSpo)
# Student Info
    print """This is the list of things that are needed
      for each of the Students on the Application.\n"""
    stuInfoList = GrabFiles('StudentInfo')
    currentAppForm.setStudentInfo(stuInfoList)
# Presentation Type
    print """This is the list of the different forms of
      Presentations that the students can select from.\n"""
    pTypeList = ChangePType('PresentationType')
    currentAppForm.setPresentationTypes(pTypeList)
# Equipment Provided
    print """This is a list of all the equipment that can
      be provided for the students\n"""
    equipList = GrabFiles('EquipmentNeeded')
    currentAppForm.setPresentationNeeds(equipList)
# Displays the information that was entered into the file
    currentAppForm.Display()
```

```
# Allows the user to change information after filling everything out
  Changing(currentAppForm, confYear)

# Pickle information
  newFile = open(confTitle1, "w")
  cPickle.dump(currentAppForm, newFile)

  #End of Program
  print """\nYou have completed the MakeTextFile program.
    The information was stored into a file called: """
  print "*** SetupData"+ str(confYear) +" ***\n"

main()
```

```
# PresentType.py
# This class is set up to keep track of the different types of presentations
#  that are allowed and the discriptions of those presentations

class PresentType:

  def __init__(self):
    self.presentType = " "
    self.discription = " "

  def __init__(self, pType, discription):
    self.presentType = pType
    self.discription = discription

  def display(self):
    print "%s: %s" %(self.presentType, self.discription)

  def setPType(self, pType):
    self.presentType = pType

  def getPType(self):
    return self.presentType

  def setDiscription(self, discription):
    self.discription = discription

  def getDiscription(self):
    return self.discription
```

```
class TagConvert:

  def __init__(self):
    self.BEGIN_ITALICS = "<i>"
    self.END_ITALICS = "</i>"
    self.BEGIN_BOLD = "<b>"
    self.END_BOLD = "</b>"
    self.BEGIN_SUP = "<sup>"
    self.END_SUP = "</sup>"
    self.BEGIN_SUB = "<sub>"
    self.END_SUB = "</sub>"
    self.ALPHA = "&Alpha;"
    self.GAMMA = "&Gamma;"
    self.DELTA = "&Delta;"
    self.THETA = "&Theta;"
    self.LAMBDA = "&Lambda;"
    self.XI = "&Xi;"
    self.PI = "&Pi;"
    self.SIGMA = "&Sigma;"
    self.UPSILON = "&Upsilon;"
    self.PHI = "&Phi;"
    self.PSI = "&Psi;"
    self.LC_ALPHA = "&alpha;"
    self.LC_BETA = "&beta;"
    self.LC_GAMMA = "&gamma;"
    self.LC_DELTA = "&delta;"
    self.LC_EPSILON = "&epsilon;"
    self.LC_ZETA = "&zeta;"
    self.LC_ETA = "&eta;"
    self.LC_THETA = "&theta;"
    self.LC_IOTA = "&iota;"
    self.LC_KAPPA = "&kappa;"
    self.LC_LAMBDA = "&lambda;"
    self.LC_MU = "&mu;"
    self.LC_NU = "&nu;"
    self.LC_XI = "&xi;"
    self.LC_PI = "&pi;"
    self.LC_RHO = "&rho;"
    self.LC_SIGMA = "&sigma;"
    self.LC_TAU = "&tau;"
    self.LC_UPSILON = "&upsilon;"
    self.LC_PHI = "&phi;"
    self.LC_CHI = "&chi;"
    self.LC_PSI = "&psi;"
    self.LC_OMEGA = "&omega;"
    self.ALEF = "&alefsym;"
    self.EMPTYSET = "&empty;"
    self.NABLA = "&nabla;"
    self.ELEMENT = "&isin;"
    self.NOTELEMENT = "&notin;"
    self.INFINITY = "&infin;"
    self.WEDGE = "&and;"
```

```
        self.VEE = "&or;"
        self.CAP = "&cap;"
        self.CUP = "&cup;"
        self.INTEGRAL = "&int;"
        self.SUBSET = "&sub;"
        self.SUPERSET = "&sup;"
        self.SUBSETEQ = "&sube;"
        self.SUPERSETEQ = "&supe;"

        self.db = {}
        self.db[self.BEGIN_ITALICS] = { "tex": "\\emph{" }
        self.db[self.END_ITALICS] = { "tex": "}" }
        self.db[self.BEGIN_BOLD] = { "tex": "\\textbf{" }
        self.db[self.END_BOLD] = { "tex": "}" }
        self.db[self.BEGIN_SUP] = { "tex": "$^{" }
        self.db[self.END_SUP] = { "tex": "}$" }
        self.db[self.BEGIN_SUB] = { "tex": "$_{" }
        self.db[self.END_SUB] = { "tex": "}$" }
        self.db[self.GAMMA] = { "tex": "$\\Gamma$" }
        self.db[self.DELTA] = { "tex": "$\\Delta$" }
        self.db[self.THETA] = { "tex": "$\\Theta$" }
        self.db[self.LAMBDA] = { "tex": "$\\Lambda$" }
        self.db[self.XI] = { "tex": "$\\Xi$" }
        self.db[self.PI] = { "tex": "$\\Pi$" }
        self.db[self.SIGMA] = { "tex": "$\\Sigma$" }
        self.db[self.UPSILON] = { "tex": "$\\Upsilon$" }
        self.db[self.PHI] = { "tex": "$\\Phi$" }
        self.db[self.PSI] = { "tex": "$\\Psi$" }
        self.db[self.LC_ALPHA] = { "tex": "$\\alpha$" }
        self.db[self.LC_BETA] = { "tex": "$\\beta$" }
        self.db[self.LC_GAMMA] = { "tex": "$\\gamma$" }
        self.db[self.LC_DELTA] = { "tex": "$\\delta$" }
        self.db[self.LC_EPSILON] = { "tex": "$\\epsilon$" }
        self.db[self.LC_ZETA] = { "tex": "$\\zeta$" }
        self.db[self.LC_ETA] = { "tex": "$\\eta$" }
        self.db[self.LC_THETA] = { "tex": "$\\theta$" }
        self.db[self.LC_IOTA] = { "tex": "$\\iota$" }
        self.db[self.LC_KAPPA] = { "tex": "$\\kappa$" }
        self.db[self.LC_LAMBDA] = { "tex": "$\\lambda$" }
        self.db[self.LC_MU] = { "tex": "$\\mu$" }
        self.db[self.LC_NU] = { "tex": "$\\nu$" }
        self.db[self.LC_XI] = { "tex": "$\\xi$" }
        self.db[self.LC_PI] = { "tex": "$\\pi$" }
        self.db[self.LC_RHO] = { "tex": "$\\rho$" }
        self.db[self.LC_SIGMA] = { "tex": "$\\sigma$" }
        self.db[self.LC_TAU] = { "tex": "$\\tau$" }
        self.db[self.LC_UPSILON] = { "tex": "$\\upsilon$" }
        self.db[self.LC_PHI] = { "tex": "$\\phi$" }
        self.db[self.LC_CHI] = { "tex": "$\\chi$" }
        self.db[self.LC_PSI] = { "tex": "$\\psi$" }
        self.db[self.LC_OMEGA] = { "tex": "$\\omega$" }
        self.db[self.ALEF] = { "tex": "$\\aleph$" }
        self.db[self.EMPTYSET] = { "tex": "$\\emptyset$" }
```

```
    self.db[self.NABLA] = { "tex": "$\\nabla$" }
    self.db[self.ELEMENT] = { "tex": "$\\in$" }
    self.db[self.INFINITY] = { "tex": "$\\infty$" }
    self.db[self.WEDGE] = { "tex": "$\\wedge$" }
    self.db[self.VEE] = { "tex": "$\\vee$" }
    self.db[self.CAP] = { "tex": "$\\cap$" }
    self.db[self.CUP] = { "tex": "$\\cup$" }
    self.db[self.INTEGRAL] = { "tex": "$\\int$" }
    self.db[self.SUBSET] = { "tex": "$\\subset$" }
    self.db[self.SUPERSET] = { "tex": "$\\supset$" }
    self.db[self.SUBSETEQ] = { "tex": "$\\subseteq$" }
    self.db[self.SUPERSETEQ] = { "tex": "$\\supseteq$" }


  def convert( self, theHTML, target):
    """ This function returns the equivalent markup in the target
    language for the supplied html.  If theHTML code or the target
    language is not supported, the original html is returned."""

    if self.db.has_key( theHTML ) :
      return self.db[theHTML].get( target, theHTML )
    return  theHTML

def main():
  myConverter = TagConvert();

  print "TeX conversions currently supported.\n"
  for key in myConverter.db:
    print "converting %s to %s" % (key, myConverter.convert( key , "tex" ))
  print "converting garbage to %s" %  myConverter.convert( "garbage" , "tex" )

  print "\nWord conversions currently supported.\n"
  for key in myConverter.db:
    print "converting %s to %s" % (key, myConverter.convert( key , "word" ))

if __name__ == '__main__' :
  main()
```

```
#    SSCA application process for Bemidji State University
#    Copyright (C) 2007  Charles Hofer

#    This program is free software; you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation; either version 2 of the License, or
#    (at your option) any later version.

#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.

#    You should have received a copy of the GNU General Public License along
#    with this program; if not, write to the Free Software Foundation, Inc.,
#    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

from Helper import *

# Used for setting up texFiles
class texBase:
  """This class contains the basic structure of a tex file for producing
  the pdf file."""

  def __init__(self):
    self.preamble = """
%-*- Mode:TeX -*-

\\newlength{\\Indent}
\\Indent = -12pt
\\def\\heading{\\hspace{\\Indent}Application Form\\hfill
 Student Presentations, Creative Performances and Poster Sessions
 \\hfill April 12, 2005 \\hfill}


\\documentclass[11pt,fleqn]{article}
\\setlength{\\textheight}{9in}
\\setlength{\\textwidth}{6.5in}
\\setlength{\\evensidemargin}{1pt}
\\setlength{\\oddsidemargin}{1pt}
\\setlength{\\topmargin}{-0.25in}

\\begin{document}

\\lineskip 5pt
\\lineskiplimit 7pt
\\topskip = 6ex

\\pagestyle{myheadings}
"""
#    annualHeader =
    self.header = pdfAnnualHeader()
    self.post = """
\\vspace{1cm}
```

```
\\noindent
Department Chair Signature\\hrulefill Date\\makebox[4cm]{\\hrulefill}

\\vspace{1cm}
\\noindent
Dean Signature\\hrulefill Date\\makebox[4cm]{\\hrulefill}


\\small{
\\section*{Dean Locations}
\\begin{description}
\\item[College of Arts and Letters]submit to Ms Tamera Mayer,
Admin.\\ Specialist, Hagg-Sauer 300
\\item[College of Professional Studies]submit to Ms Joan Miller,
Admin.\\ Specialist, Decker 110
\\item[College of Social and Natural Sciences]submit to Ms Colleen
Jennings, Admin.\\ Specialist, Sattgast~232
\\end{description}
}

\\small{\\emph{Support is provided, in part, by the College of Arts and
Letters, the College of Professional Studies, the College of Social and
Natural Sciences, the Honors Program, the Alumni Association, the
University Foundation, the Student Senate, the Office of Academic and
Student Affairs, the Office of the President, the Graduate Student
Association, and the Center for Professional Development.}}
\\end{document}
"""
```

```
#!/usr/bin/python

#    SSCA application process for Bemidji State University
#    Copyright (C) 2007  Charles Hofer

#    This program is free software; you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation; either version 2 of the License, or
#    (at your option) any later version.

#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.

#    You should have received a copy of the GNU General Public License along
#    with this program; if not, write to the Free Software Foundation, Inc.,
#    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

#!/usr/bin/python
import os
import cgi
import time
import cPickle
import string
from Helper import *
from AppForm import *

# Used if there was a problem on the application form
def fixProblem( problem, firstStudent ):
  printHeader( "There was a problem with your application" )
  print """<p>You failed to supply %s in your application.
           Use your browser's back button to re-enter the appropriate
"          information.""" % problem
  print firstStudent
  print "</body></html>"

# Checks for errors on the application form
def allRequiredFieldsPresent(pairs, firstStudent):
  if not pairs.has_key(firstStudent):
    fixProblem("at least one presenter", firstStudent)
  elif not pairs.has_key("title"):
    fixProblem("a title", firstStudent)
  elif not pairs.has_key("abstract"):
    fixProblem("an abstract", firstStudent)
  elif not pairs.has_key("presType"):
    fixProblem("a contribution type", firstStudent)
  elif not pairs.has_key("sponsorFName1"):
    fixProblem("a faculty sponsor", firstStudent)
  elif not pairs.has_key("sponsorLName1"):
    fixProblem("a faculty sponsor", firstStudent)
  else:
    return True
```

```
def addMissingPairs(pairs, studentNum, itemList):
  for count in range(1,len(itemList)):
    if not pairs.has_key(itemList[count] + str(studentNum+1)):
      pairs[ itemList[count] + str(studentNum+1)] = ""


def addMissingSponsorPairs(pairs, sponsorNum):
  if not pairs.has_key("sponsorFName" + sponsorNum):
    pairs[ "sponsorFName" + sponsorNum ] = ""
  if not pairs.has_key("sponsorLName" + sponsorNum):
    pairs[ "sponsorLName" + sponsorNum ] = ""
  if not pairs.has_key("sponsoremail" + sponsorNum):
    pairs[ "sponsoremail" + sponsorNum ] = ""
  if not pairs.has_key("sponsorphone" + sponsorNum):
    pairs[ "sponsorphone" + sponsorNum ] = ""

# Page after application for students to verify all information
def buildPage( pairs, fileBase, currentAppForm):
  maxStudent = currentAppForm.getMaxStudent()
  maxFaculty = currentAppForm.getMaxFaculty()
  stuList = currentAppForm.getStudentInfo()
  stuInfo = []
  for item in stuList:
    stuInfo.append(item[0:len(item)-1])

  printHeader( "Verify Application Form" )

  print """<p><span style = "font-weight: bold">
              Please review this information carefully.  It is your
"             last chance to change any of the information on this
"             application.
"             </span></p>"""
  print """<p>Title: <span style = "font-weight: bold">%s</span></p>""" \
      % cgi.escape ( pairs [ "title" ] )

  abstract =  pairs [ "abstract" ]
  abstract = abstract[:1200]
  print """<p>Abstract: %s</p>""" \
  % cgi.escape ( abstract )

  for i in range (0, int(maxStudent)):
    if pairs.has_key(stuInfo[0] + str(i+1)):
      addMissingPairs( pairs, i+1, stuInfo )
      print """<p><span style = "font-weight: bold">
      Presenter %s:<br> </span> Name: %s %s <br>""" \
        % (str(i+1), cgi.escape ( pairs [ stuInfo[0] + str(i+1) ]),
"                   cgi.escape ( pairs [ stuInfo[1] + str(i+1) ]))

      for count in range(2, len(stuInfo)):
        print stuInfo[count] + """: %s <br>""" \
            % cgi.escape ( pairs [ stuInfo[count] + str(i+1) ] )
      c = str(i+1)
      print "Student Type: %s"\
          %(cgi.escape ( pairs [ "stuType" + c ]))

 contribType = pairs [ "presType"]

 print """<p>Your application indicates your contribution is a:
        <span style = "font-weight: bold">%s</span></p>""" \
        % cgi.escape ( contribType )

 if pairs.has_key("equip" ):
   print """<p>You have indicated the following equipment needs:
   <span style = "font-weight: bold">""",
   theList = pairs[ "equip" ]
   for item in theList :
     if len(theList) == 1:
       print """ %s.""" %cgi.escape ( item )
     elif item == theList[len(theList)-1]:
       print """and %s.""" % cgi.escape ( item )
     else:
       print """ %s,""" %cgi.escape ( item )
   print "</span></p>"

 if pairs.has_key("specialNeeds"):
   print """<p>You have indicated the following special needs:
          <span style = "font-weight: bold"> %s </span></p>""" \
          % cgi.escape ( pairs[ "specialNeeds" ] )

 for i in range (int(maxFaculty)):
   c = str(i+1)
   if pairs.has_key("sponsorFName" + c):
     addMissingSponsorPairs( pairs, c)
     print """<p>Faculty Sponsor: %s %s """ \
       % ( cgi.escape ( pairs [ "sponsorFName" +c ] ),
       cgi.escape ( pairs [ "sponsorLName" +c ] ))
     print """<br>Faculty Sponsor Email: %s""" \
       % cgi.escape ( pairs [ "sponsoremail" +c ] )
     print """<br>Faculty Sponsor Phone: %s </p>""" \
       % cgi.escape ( pairs [ "sponsorphone" +c ] )


 print """
 <p>
 <form method = "get" action = "Keepit.py"  >
 <input type=hidden name=trackingNumber value=%s>
 <input type = "submit" value = "Submit application" />
 Electronically submits your application and
 generates a form for you to print and submit.
 Note that if you used any mark up codes for special characters, the
 codes will be converted to the special characters in this step.
 </p>
 </form>

 <p>
 <input type="button" value="Change Information"
```

```
  onclick="history.go(-1)" />
  if any of the values are incorrect.
  </form>
  </p>

  <p><form method = "get" action = "%sselect.html">
  <input type = "submit" value = "Start Over" />
  to get a new blank form.
  </form>
  </p>""" %(fileBase, selectDirectory())


  print "</body></html>"

# Puts information in a dictionary for retrieval by Keepit.py
def fs2Dict( pairs ):
  dict = {}
  dict["equip"] = []
  for item in pairs.value:
    if  item.name == "equip":
      dict["equip"].append(string.replace(item.value, "\r\n"," "))
    else:
      dict[item.name] = string.replace(item.value, "\r\n"," ")
  return dict

# Pickles the dictionary
def pickleDict( dict, fileBase ):
  directory = wwwDirectory() + "proposals/"+fileBase
  fileN = open (directory, 'w')
  cPickle.dump( dict, fileN)
  fileN.close()

def main():
  if (os.path.exists ("setUpData")):
    fName = open("setUpData", 'r')
    currentAppForm = cPickle.load(fName)

    stuInfo = []
    stuInfo = currentAppForm.getStudentInfo()
    stuInfoFirst = stuInfo[0]
    stuInfoFirst = stuInfoFirst[0:len(stuInfoFirst)-1]
    firstStudentInfo = stuInfoFirst + "1"
    pairs = cgi.FieldStorage()

    if allRequiredFieldsPresent(pairs, firstStudentInfo):
      base = str(int(time.time()) % 1314000 )

      name = string.split ( pairs[ firstStudentInfo ].value )
      fileBase = name[0] + base

      dict = fs2Dict( pairs )
      pickleDict( dict, fileBase )
      buildPage(dict, fileBase, currentAppForm)
```

```
main()
```