ABSTRACT

In this project, we present the Interactive Survey Development Kit (ISDK), an enhanced tool that extends the survey structure developed within the Computer Science department to support data collection. Compared with the current setup, maintained by Dr. Carl Bredlau using Java on a Solaris platform, and focused on survey administration and data collection, ISDK is developed as platform independent and can be seen as the survey design tool. It allows the user to generate their own surveys by providing functionalities that include input of survey information either through a text file that can be uploaded or as a sequence of survey items entered one by one, format with images, saving and retrieval of partially developed surveys as well as survey preview. Once the survey is created, a zip archive containing all the necessary files is created which can be sent to the administrator in order to deploy the survey.

The project involved the design and implementation of ISDK as an attractive way to automate the survey creation mechanism currently being used within the Computer Science department. Coupled with the survey management application, ISDK can be seen as a robust survey design and management environment that can be distributed through the CS webpage to interested parties, enhancing the department's image as high quality education environment.

INTERACTIVE SURVEY DEVELOPMENT KIT

(ISDK)

by
SHILPA VENUGOPAL

A MASTER'S PROJECT
Submitted in partial fulfillment of the requirements
For the degree of Master of Science in
The Department of Computer Science in
the Graduate Program of
Montclair State University
May 2006

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Stefan Robila, for giving me the opportunity to work on this project. I want to convey my deepest appreciation for his support and invaluable guidance throughout the project. I have yet to see the limits of his wisdom, patience, and selfless concern for his students. Without his guidance and persistent help this project would not have been possible.

Words are inadequate to express my gratitude to Dr. Carl Bredlau for all his help and constant feedback In particular; Dr. Bredlau's recommendations and suggestions have been invaluable for the project.

I would like to take this opportunity to thank the graduate committee members for their time and effort.

Finally, it is to my husband, Santosh Satyan, that I owe the most overwhelming debt of gratitude for his encouragement and support.

TABLE OF CONTENTS

TABLE OF FIGURES

1. Introduction

Accurate and ongoing feedback is imperative for success in any environment. Surveys are an ideal mechanism to gather and analyze large amounts of direct feedback about characteristics, behaviors, opinions and knowledge of a particular population. No matter what the subject matter, all surveys are conducted to gather information relevant to a specific problem or situation. Something as simple as a web survey can give you the strategic intelligence to make better decisions quickly, easily and more cost effectively.

For those who have internet-enabled audiences, nothing is faster or more cost-effective than a web-based survey [1]. A web-based survey can be developed and deployed in a fraction of the time of other survey solutions and results can be tabulated in real-time. Web-based surveys, in comparison to telephone and mail surveys, provide equivalently valuable information less expensively, more quickly and often result in significantly higher response rates. Online, web-based surveys provide an efficient data collection option for respondents with access to e-mail and/or the World Wide Web. These surveys streamline efforts by eliminating the need for postage, faxes, e-mail attachments and/or manual tabulations of results (thus reducing data entry errors). Their ease of use also increases response rates and provides instant turnaround of responses. Current studies also suggest that there are no significant differences in the way respondents approach a web or paper based survey and that the answers provided through one or the other are comparable [2]. In addition, topics such as computer and information technology seem to be more appropriate for web-based surveys ([2], [3]).

These studies are of particular interest to academic institutions such as MSU that use surveys to gather student feedback for evaluating teaching and for assessing student knowledge. Within the last years, the Department of Computer Science at MSU has embarked on several initiatives that required extensive collection of feedback from the students. For this, faculty members must be able to create, administer and analyze surveys in a quick, easy-to-use and efficient manner. The department has created a survey system that is versatile and convenient but the current state of the art requires the person

maintaining and updating the system to perform significant work in file creation and editing.

This project involves the design and implementation of an Interactive Survey Development Kit (ISDK) that extends upon the survey structure initially developed within the Computer Science department to support data collection. ISDK is designed to be an easy to use, flexible survey creation tool that enables the user to create surveys quickly and easily using a completely point-and-click user-interface. The user has complete control over the look and feel of his or her survey.

This report is organized as follows. In the next section we discuss the requirements for the application together with a discussion of the new functionality introduced. Section 3 provides a summary comparison of ISDK and other survey tools usually available to academia. The design of the application as well as implementation issues are discussed in Section 4. An extensive description of the usage of the application is presented in Section 5. The report ends with conclusions (Section 6), references and a full listing of the code. We note that the code is also included in the attached CD.

2. Requirements

2.1. Survey Design Application

Within the last years, the Department of Computer Science has embarked on several initiatives that required extensive collection of feedback from the students. Examples of this are the survey supporting the IT degree proposal and the assessment surveys administered for the 109 classes (at the beginning and end of Spring 2005 and the beginning of Fall 2005 semester) as well as the surveys for each of the core undergraduate CS courses. Until now, the survey mechanism was developed and maintained by Dr. Carl Bredlau [4]. While extremely versatile and convenient, the current state of the art requires the person maintaining and updating the system to perform significant work in file creation and editing.

In order to create a survey, a user provides a text file with questions that is manually adapted to JSP and HTML. The process requires careful user supervision. First, it is necessary to ascertain that the questions are aligned such that each question is on a separate line and that they do not have any question numbers or bullets (these must be removed). In order to create the HTML and JSP, it is necessary to first determine the question type to see if it is a multiple choice question, single line text type of a question or an essay type question. Once this is determined, the question type tags are appended before each question such that the tags and questions are tab delimited. The question type tags currently used are:

- [MC] for multiple-choice questions,
- [TXT] for single line text type of a question
- [ESS] for an essay type question.

Once the tags are appended, the next step is to determine the answer choices for the multiple choice questions. Every question has to be analyzed to find out if the answer choices would be Yes/ No, Strongly Agree/ Agree/ Neutral/ Disagree/ Strongly Disagree, High/ Moderately High/ Medium/ Moderately Low/ Low etc. Next, the answer choices

are appended at the end of the corresponding question using tab delimiters. Figure 1, presents an example of such a file as seen by a user in Textpad.

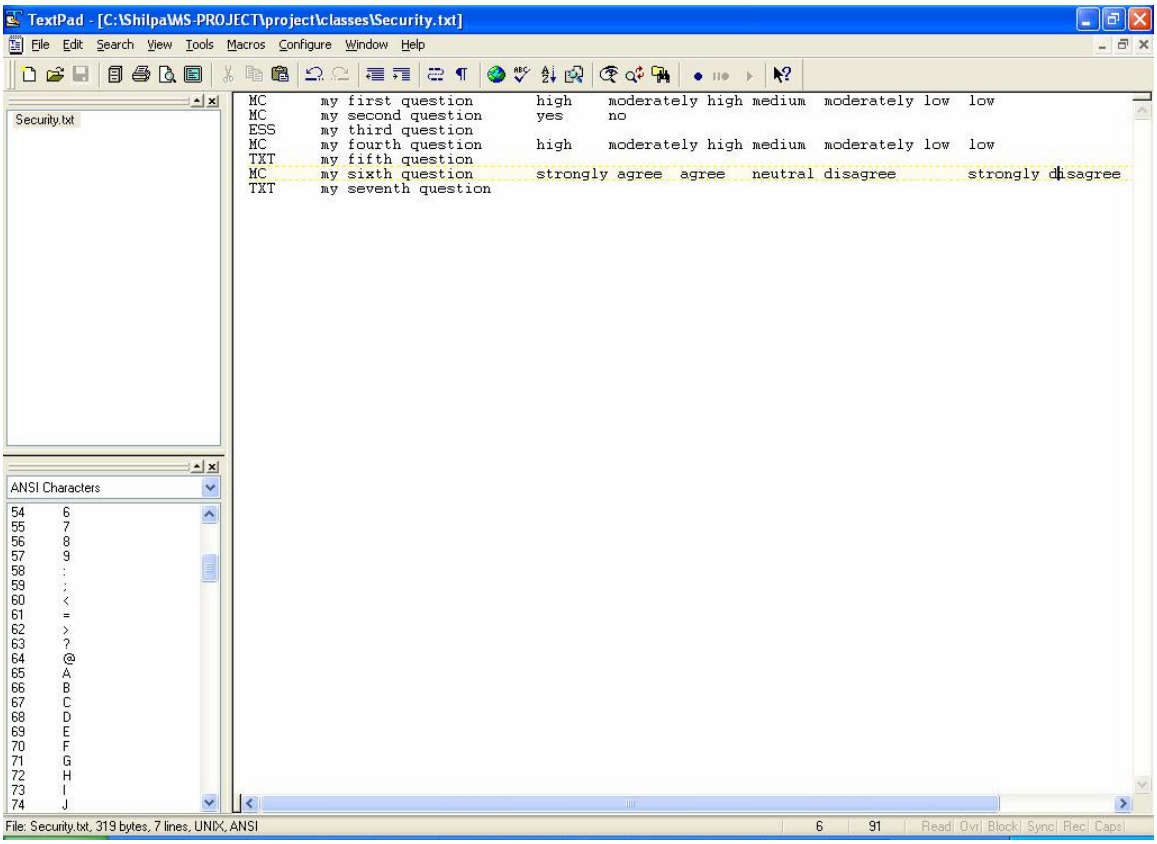


Figure 1 - Example of a "tagged file"

The resulting text file is used as an input to a Java program called Makeform.java. When this program is executed successfully, the corresponding HTML file is generated.

An existing template is used to create the JSP file. The modifications that have to be made to the template include changing the course number, course name, course description, instructor name and section number. If there is more than one section for that particular course, suitable code has to be added to reflect the corresponding instructor names and section numbers. Once these changes have been made and saved, the corresponding JSP file is created.

The three files thus created namely, the formatted text file, the HTML and the JSP file have to be sent to Dr. Bredlau so that the survey files can be hosted on Tolstoy (i.e. tolstoy.montclair.edu).

The purpose of ISDK is to automate this entire process such that the user can create the survey files, at the same time, ensuring transparency of the file formatting.

2.2. New Functionalities

Interactive Survey Development Kit (ISDK) extends upon the survey structure initially developed within the Computer Science department to support data collection. The current system requires the user to send the text file containing all the questions to the person performing the manual transformation. At any point, if the user wants to modify the questions or add any question, he or she would have to send a new copy of the text file. As opposed to that, ISDK allows the user to generate his or her own surveys, by providing new features. Below we describe several that were part of the project requirements:

- ***"Upload Survey Questions"*** allows the user to browse through the system and choose a text file containing the questions that can be uploaded. When the user clicks on this option, the "***Choose Input Survey File***" dialog box opens. Once the user selects a text file and clicks on Open button, the questions in the text file are uploaded onto the screen.

- "***Blank Survey"*** allows the user to add questions manually. The screen initially displays a single question slot containing the message "*Type your question here*". Once the user has typed a question in this slot, she/he can add any more questions by clicking on the "*Add question*" button.

- ***"Add Question"*** allows the user to add questions manually. The user can either add a question at the end or at any specific position. In order to add a question at the end, the user clicks on the button and the question will be added at the end. If the user wants to add a question at any specific position, the user has to select the question before which he wants to add the new question and then click on the "Add question"

button. The new question is inserted before the question that was selected. While inserting questions at specific positions, the user can add multiple questions at once by selecting multiple positions and when he clicks on the “Add question” button, new questions would be added before all the questions that were selected. Once the user has provided all the questions, the questions are presented in a format that allows the user to perform all the tasks necessary to create the actual survey files. The questions will appear in the same order as the text file or the order in which the user entered the questions and every question is assigned a question number. Every question number has a drop-down menu that lets the user change the order of the questions if he wishes to do so.

- ***“Rearrange Questions”*** allows for the rearrangement of the survey items. There are two ways that the user can choose to rearrange the questions:
  1. Check the box next to the question and choose a new position for the question. When this action is performed, the questions are automatically rearranged.
  2. If more than one question needs to be rearranged, the user can choose the new positions for all the questions that need to be changed. Once this is completed, the user can click on the "Rearrange Questions" button or the "Rearrange Questions" options under the "Actions" menu at the top.

  The changes will be reflected immediately and the questions will appear in the updated order. If the user wishes to change the questions, he can do so by editing the questions in the GUI rather than having to change them in the text file. In order to accommodate questions of any length, scroll bars have been provided so that the user does not have to restrict the length of the question.

- ***“Remove question”*** allows the user to delete either a single question or multiple questions. The user has to click on the question that he wants to delete and click on the "Remove question" button.

ISDK provides flexibility to the user in selecting the answer choices for the questions. The user can either select answer choices from a list of standard answer types

or can provide his or her own answer choices. The following requirement option addresses the selection of question type

– ***"Choose Answer Type"*** lets the user select answer choices for the questions from a drop-down menu containing the following standard answer choices
    - **Yes / No**
    - **High / Moderately High / Medium / Moderately Low / Low**
    - **Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree**
    - **Text**
    - **Essay**
    - **True / False**
    - **Ok / Not Ok**

If the above answer choices are not suitable for the questions, the user can provide his own choices by using a feature called ***"Add your own answer type"*** that lets the user create his own answer choices. In order to create a new answer type, the user has to first enter a short description (less than 10 characters) for the answer type which will be displayed in the drop-down menu. He has to then type the actual answer choices one on each line in the text box called "Answer choices". There is a feature called ***"Create new type"*** that allows the user to add this new type to the list of standard types so that the next time he wants to use these choices, they are available rather than having to re-create it each time.

Once the appropriate answer choices are available, the user can assign an answer type to each question by selecting the question and then choosing the appropriate answer type from the drop-down menu. Once he has selected the answer type, clicking on "Apply changes" button will append the answers to the questions and the corresponding answer choices are displayed next to the question so that at any point, the user can keep track of the assigned answer types. If the user wishes to change the answer type for any question, he can always do so by choosing a different answer type from the drop-down menu. If multiple questions have the same answer choices, in order to save the user some time in selecting the same answer type for each question, check boxes have been

provided next to each question such that the user can select multiple questions at a time and select the answer type once from the drop-down menu.

ISDK has been designed to support images for surveys. The following two options were included in the requirements to address this:

- ***"Choose Image"*** allows the user to append an image to any question. The user can attach an image to a single question or to multiple questions at once. The user has to select the questions for which he wants to append a particular image and once he clicks on the "Choose Image" button, a dialog box opens. The user can only select files with extension .gif, .jpg, .jpeg, .bmp and .tiff. Once the user has selected the image, he has to click on the "Apply changes" button for the images to be appended to the particular question. Clicking on the "Apply Changes" button displays the pathname of the image file just below the question. The actual image can be viewed by clicking on the "Preview" button. At any point, the user can also delete images from a survey.

- ***"Remove Image"*** allows the user to delete an image that has been appended to a question. The user can either delete a single question or multiple questions at once. The user has to select the questions for which the images have to be deleted and then click on "Remove Image" button to delete the images from the questions.

ISDK also provides a feature that allows the user to quit the application half-way through and resume it at a later time from the point where he left off:

- ***"Save Survey"*** allows the user to save all the information about the survey including all the questions and the answer choices that have been selected so far along with any images so that he does not have to start all over again. In order to save a survey, the user has to first enter the header information such as survey name, course name, course description, professor name and section. This information will be dynamically retrieved while creating the actual survey files. The survey name field is a mandatory field and should preferably not end with an underscore in which case it may affect the file name formats for images.

- ***"Upload Past Survey"*** allows the user to select a survey that he was previously working on and resume it from the point where he left off. Past survey files also include completed surveys for which the survey files have already been generated. The user can retrieve a previously completed survey and make any modifications to it.

The user does not have to wait for the survey files to be created in order to see how the actual survey looks:

- ***"Preview"*** allows the user to view the survey page even before creating the actual survey. This feature helps the user in making modifications before creating the survey. In order to preview the survey, it is not necessary for the user to have completed working with all the questions. If the user has selected answer choices only for some of the questions, in the preview the answer choices for the remaining questions will show up as unknown.

Finally, the ***"Create Survey"*** feature is the main focus of the application. It is used to create the actual survey files namely the formatted text file, the HTML file and the JSP file. Once the survey files are created, a confirmation message appears on the screen specifying the location of the survey files. For every new survey, a new folder having the survey name is created within which the survey files are stored. A zipped file containing all the survey files are also created that can be sent to the administrator to deploy the survey.

3. Comparison with other Survey Systems

3.1. MSU Survey System

This system [5] is a web-based tool which enables end users to autonomously create and run online surveys, feedback or registration forms. Its built-in user management system facilitates collaborative survey development and administration. It is based on an Open Source application developed at Virginia Tech [6]. The following are some of the differences between the MSU Survey System and ours:

a. MSU System provides user authentication via internal user database (XML-based) and/or external via LDAP. It provides online storage allowing a user to access the survey from any place. In contrast, our system stores the surveys locally and does not require network or internet access. Authentication for tolstoy.montclair.edu or direct access for survey placement can be implemented but was not within the goals of this project. In terms of local versus online storage, each has its own benefits depending on the type of environment.

b. MSU System allows the introduction of one question at a time, similar to our approach. But our system also provides the option of uploading a text file containing all the questions such that the user does not have to manually enter the questions. The advantage is that if the user creates multiple surveys containing common questions, he or she does not have to enter the same question multiple times.

c. MSU System does not provide image handling. ISDK, on the other hand allows the inclusion of images along with survey questions. The user can browse through the system and attach any image whose format is supported by the system. At any point, the user can also delete images. Images can be viewed in the preview so that the user can get an idea of how the image looks in the actual survey.

d. With MSU System, in order to relocate a question, it has to be moved multiple times depending on the new location. A question can either be moved above or below at a time. For example, if the first question has to be moved to the sixth position, it has to be moved below five times. ISDK allows a question to be relocated to any position in one step just by specifying the new position for the question.

e. MSU System does not allow multiple questions to be rearranged at a time. In case multiple questions need to be rearranged, it can only be done one question at a time. ISDK allows multiple questions to be relocated in one step.

f. MSU System requires the user to specify the answer choices at the time of constructing the question. It does not allow the user to select answer choices at a later time. ISDK provides flexibility to the user in selecting answer choices at any time even several days apart.

g. MSU System requires the user to enter answer choices separately for every question. If multiple questions have the same answer choices, the user would still need to enter them each time for every question. ISDK allows the user to specify answer choices at once for multiple questions that have the same answer choices.

h. MSU System does not allow the user to save answer choices for future use. Each time an answer choice is to be used, the user has to enter them from scratch. ISDK provides flexibility by allowing the user to create answer choices once and they are retained for future use.

i. MSU System does not allow the user to add / delete multiple questions at once. ISDK provides both these functionalities.

j. MSU System usage is restricted to people who have a netID whereas ISDK can be used by anyone who has access to the application.

k. MSU System allows the user to format the survey with graphical features such as fonts and colors. Our system does not include these features at present but can be added as an extension of our system.

l. MSU System allows the user to insert text in between questions. Our system does not provide this flexibility.

m. MSU System allows for results handling in a flexible manner (including csv, SAS and SPSS format). Our system does not provide for results handling but can be added later by new projects. This was not within the goals of the project and relates more to survey management than survey design.

3. 2. Blackboard Survey System

Blackboard is a commercial application software sold and maintained by the e-Learning company Blackboard Inc. [7]. The Survey Manager [8] is part of the course management suite and allows the user to create anonymous, non-graded surveys. It also allows the user to modify and remove surveys from any Blackboard course. All surveys created for the course are located in the Survey Manager, no matter what content area is used to deploy the survey. In surveys, correct answers are not identified and a statistical analysis of the answers is provided. This feature can be used for course or instructor evaluations, or to gather demographical information. Instructors can use the Survey Manager to guide course curriculum by asking students questions on pacing, the need for clarification, etc. The following are some of the differences between our system and the Bb Survey Manager:

a. Every survey that is created belongs to a course. Access to the surveys is limited only to a single class and only to users registered in the class. ISDK on the other hand does not present such restrictions. Survey taker authentication, while possible in future implementations of the CS survey system was not part of the ISDK requirements.

b. Blackboard Survey System provides online storage allowing a user to access the survey from any place. In contrast, our system stores the surveys locally and does not require network or internet access.

c. Blackboard Survey System does not provide image handling. ISDK, on the other hand allows the inclusion of images along with survey questions. The user can browse through the system and attach any image whose format is supported by the system. At any point, the user can also delete images. Images can be viewed in the preview so that the user can get an idea of how the image looks in the actual survey.

d. Blackboard Survey System does not allow multiple questions to be rearranged at a time. In case multiple questions need to be rearranged, it can only be done one question at a time. ISDK allows multiple questions to be relocated in one step.

e. Blackboard Survey System requires the user to enter answer choices separately for every question. If multiple questions have the same answer choices, the user would still need to enter them each time for every question. ISDK allows the user to specify answer choices once for multiple questions that have the same answer choices.

f. Blackboard Survey System does not allow the user to save answer choices for future use. Each time an answer choice is to be used, the user has to enter them from scratch. ISDK provides flexibility by allowing the user to create answer choices once and they are retained for future use.

g. Blackboard Survey System does not allow the user to add / delete multiple questions at once. ISDK provides both these functionalities.

h. Blackboard Survey System is managed through a third party vendor and requires financial investment. ISDK is in house developed.

i. Blackboard Survey System provides a spell-check feature that is not supported by our system.

j. Blackboard Survey System allows the user to specify if the question should be in plain text, smart text or html. Our system does not provide this flexibility.

4. Application Design

Figure 2 presents the class diagram for ISDK. In the following, we discuss some of these classes in detail.

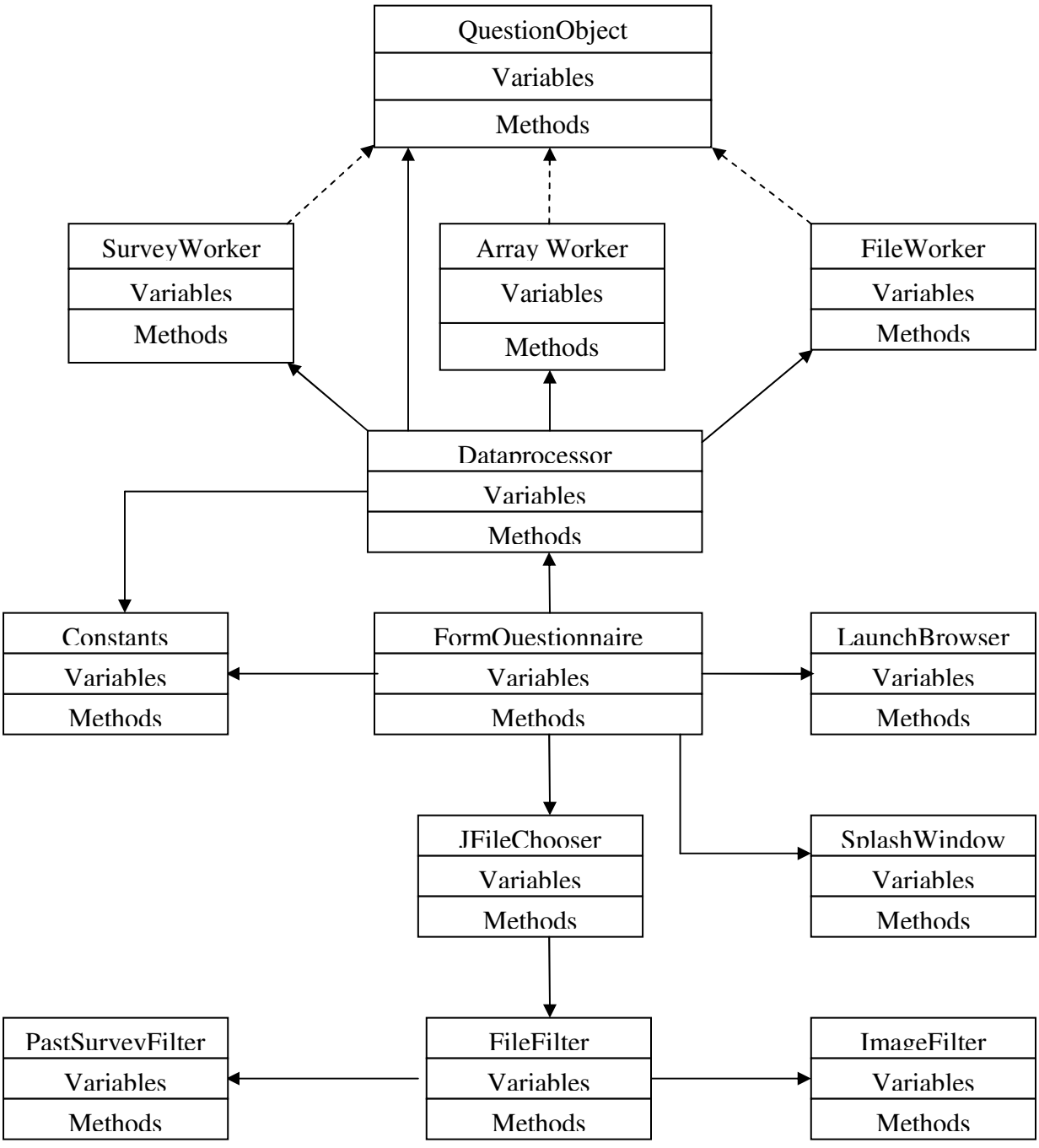


Figure 2 - Class diagram for ISDK

4.1. DataProcessor Class:

This class is an interface between the different workers in the application, i.e. FileWorker, ArrayWorker and SurveyWorker. It provides a level of abstraction of the actual process to the class that invokes the functionality. These workers are designed to complete specific tasks. This class performs the requested operation with the help of the appropriate worker or by using a combination of different workers. It is designed to generate the MC formatted file for the given list of questions (along with the question attributes). It is also used to read the contents of the MC formatted file and return a list of question objects to the calling entity. Figure 3 is the class diagram representation of the DataProcessor class.

4.2. FileWorker Class:

This class is used to read the contents of the given file and return a list of strings, each representing the different lines within the input file. It extracts all the question objects from the list of questions and creates the MC formatted file in the format described earlier. Figure 4 is the class diagram representation of the FileWorker class.

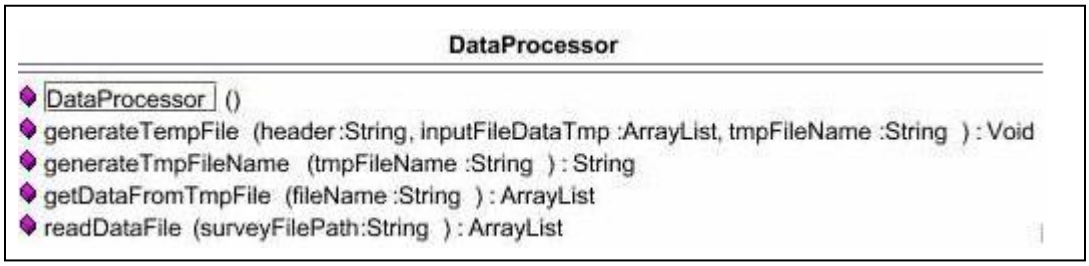


Figure 3 - Data Processor Class diagram

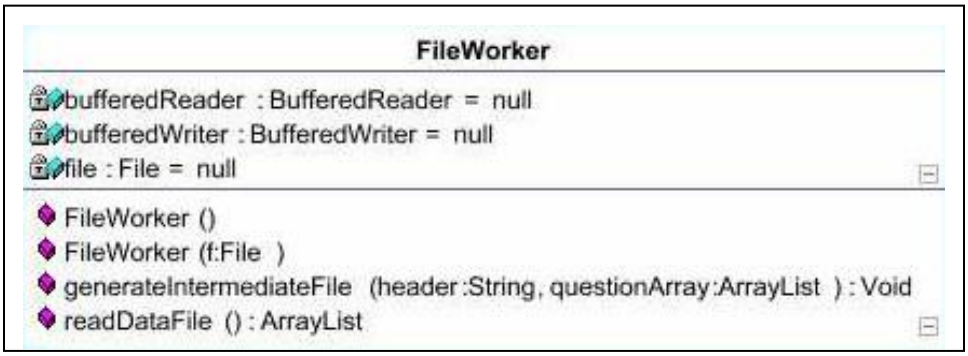


Figure 4 - FileWorker Class diagram

4.3. SurveyWorker Class:

This class is designed to generate the survey files. The survey files generated are used both for preview purpose as well as the final distribution file. The final distribution file contains the HTML file, JSP file, any images that were attached to each question and the text file in the format that is consistent with the current system. The look-and-feel of the generated HTML file and the JSP file can easily be changed by changing the template that is used to create these files. Two files, namely the JSPTemplate.jsp and HTMLTemplate.txt are used as templates to generate the JSP file and the HTML file for all the surveys. These files contain the generic information that is common across all the surveys and placeholders that are specific to each survey. When these files are generated specifically for a survey, the placeholders are replaced with the appropriate values specified by the survey creator. This class provides methods to copy the images from the users file system into survey specific folders generated on-the-fly and also to generate the survey file. As a convenience feature, this class also creates a zip file containing all the generated files. Figure 5 is the class diagram representation of the SurveyWorker class.

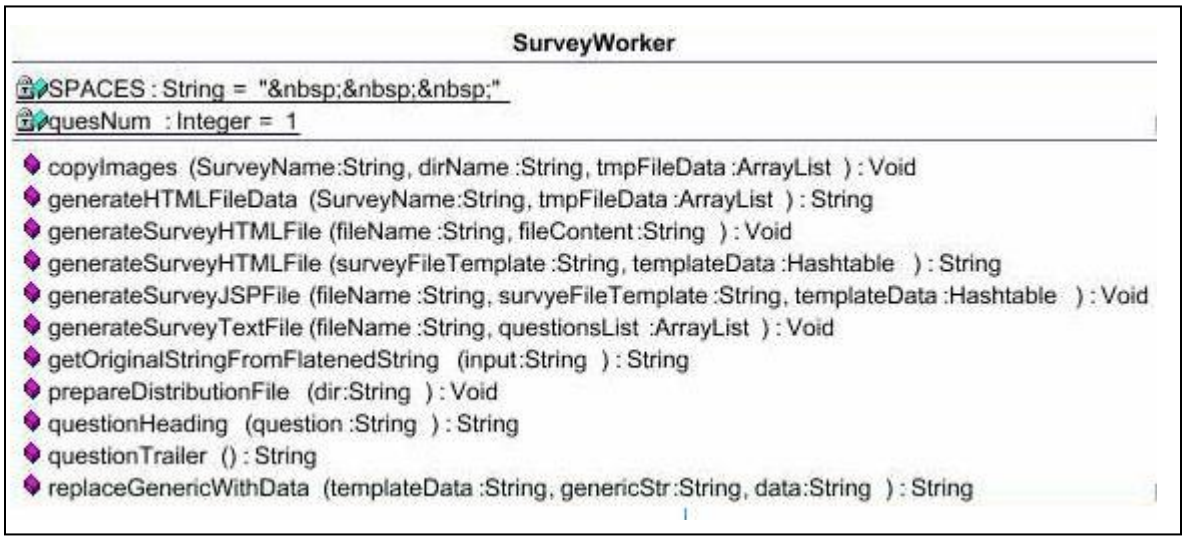


Figure 5 - SurveyWorker Class diagram

4.4. ArrayWorker Class:

This class works with a list of strings and parses it to create a list of question objects. According to the format of the MC formatted file, each attribute of the question is separated by a “tab”. It looks at each string (representing the question along with all the attributes that it contains at that instant) and tokenizes it using ”tab” as a delimiter. For each of the questions in the list, new question objects are created. After tokenizing, each token is then used to set appropriate question attributes. All such question objects are then returned to the caller as a list of question objects. The attributes of the question object change over time and are reflected in the MC formatted file when the user chooses to save the survey. Questions can be persisted as a completed question or as an incomplete question. A question is considered to be complete when it contains an answer. Incomplete questions are represented by “-1” for the question type. Figure 6 is the class diagram representation of the ArrayWorker class.

4.5. LaunchBrowser Class:

This class contains two static methods – one to determine the OS type and the other to launch the user’s default browser. The method that invokes the default browser (if it is a windows OS) accepts the path to the file and launches the browser. Figure 7 is the class diagram representation of the LaunchBrowser class.

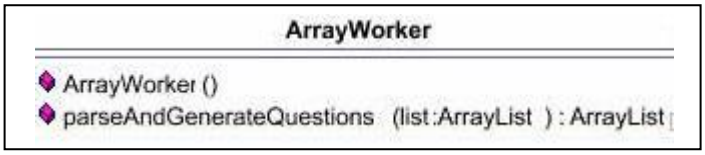


Figure 6 - ArrayWorker Class diagram

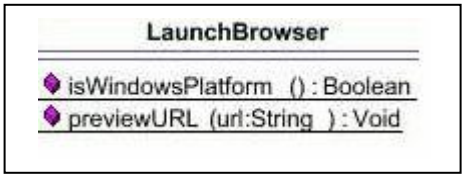


Figure 7 - LaunchBrowser Class diagram

4.6. QuestionObject Class:

This class represents the questions and defines all the attributes (question, question type, answers and the associated image name) of a question. Each question is displayed on the screen (and ultimately on the survey) in a certain order. However, the position of the question in the survey is not captured in the attributes of the question. Instead, this is dynamically determined by looking at the position within the data structure (or the list). Figure 8 is the class diagram representation of the QuestionObject class.

4.7. SplashWindow Class:

This class loads the initial credits on application startup and also when the "About" file menu option is clicked. The class that invokes this class listens to a variable that toggles between two values. The state of this variable changes when the user clicks on the window. Also, at this instant, the window is disposed from memory. On this state change, the class that loads the window continues execution. Figure 9 is the class diagram representation of the SplashWindow class.

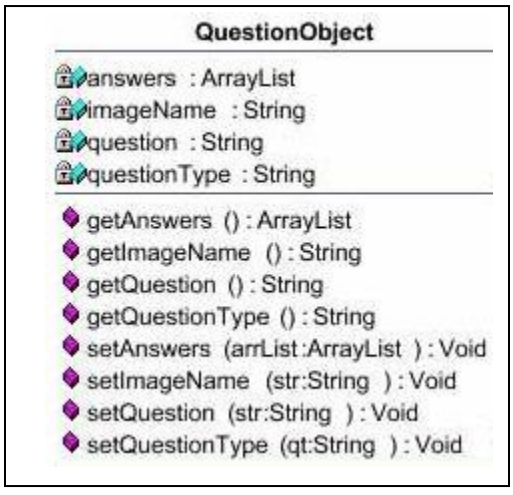


Figure 8 - QuestionObject Class diagram

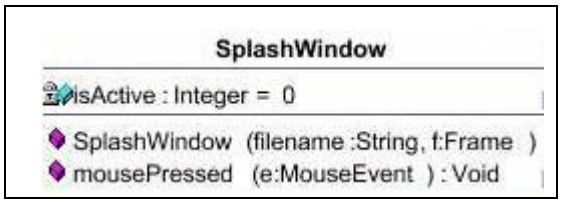


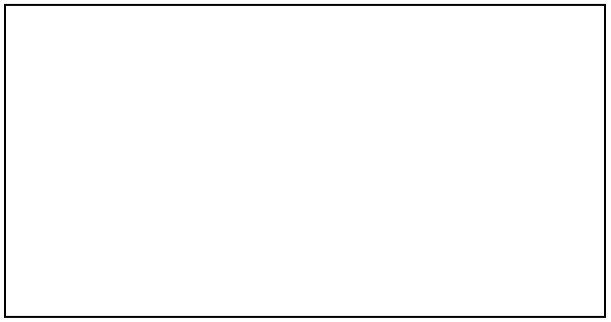
Figure 9 - SplashWindow Class diagram

4.8. Constants Class:

This is the class that initializes the data-structures that are utilized by the entire application. It loads all the application properties and the standard answer choices into corresponding data-structures when the class loads for the first time. The data-structures are declared to be static since all the objects of this class and other classes need to share the same set of values at any given time. It loads information from "application.properties" file and "constants.properties" file that holds all the application-specific configuration parameters and the standard answer choices. The application properties' information varies from the file extension of the temporarily generated file to the colors that the users see on the screen on application startup. Figure 10 is the class diagram representation of the Constants class.

4.9. FormQuestionnaire Class:

This is the main class that controls all the other classes and also the generation of the ISDK GUI. All actions that the user performs fire appropriate triggers that are handled by different listeners. In response to the trigger, this class responds to the action either internally or in turn uses other classes to complete the requested operation. It provides the GUI elements to aid the user in creating the survey – ability to enter survey header information, specify question attributes, add/remove questions, rearrange questions, preview the survey, create new custom answer types and finally create the survey. Any changes to the question's attributes are reflected on the GUI immediately. These changes are not persisted until the user specifies it and the changes are kept only in memory. The various functionalities have been designed as follows:



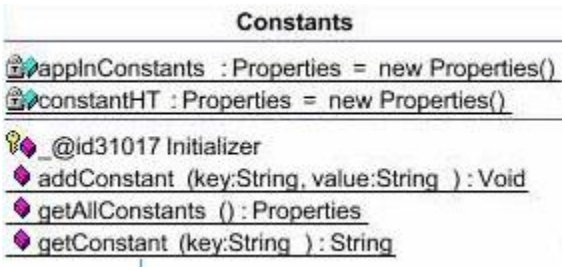


Figure 10 - Constants Class diagram

- **Upload Survey**: This provides a means to upload a new survey from a file. This uses *DataProcessor* to read the contents of the file and populates the questions on the GUI.
- **Add Questions**: This provides the user with the ability to add a question at the end or at any desired position(s). When a new question is added, new GUI elements are added to address the question's different attributes. If the question was not added at the end, all the questions below the desired position are moved one position below along with all the attributes of the question. In addition to the changes on the screen, the different data structures that hold the question's attributes are also updated.
- **Remove Questions**: This provides the user with the ability to delete a question at any desired position(s). If the question that was deleted is not at the end, all the questions below the desired position are moved one position above along with all the attributes of the question. In addition to the changes on the screen, the different data structures that hold the question's attributes are also updated.
- **Choosing an image**: When the user selects an image for a question, the name of the image is assigned to a variable. Only when the user clicks on "Apply Changes", the value gets stored in the data structure in memory.
- **Removing an image**: If multiple images are deleted at once, all the questions for which the images need to be removed are added to a list and when the user clicks on "Apply Changes" these values are removed from the data structure.
- **Apply Changes**: This is used to determine if an answer type or an image was chosen for a question and the corresponding information is stored in the appropriate data structures.

- **Save Survey**: It reads all the questions and the header information from the screen, combines this information with the information stored in the data structures about the answer types and the image associated with the questions and finally uses DataProcessor class to persist the information to an MC formatted file.
- **Upload Past Survey**: This provides a means to upload a past survey from a file. It uses *DataProcessor* class to read the contents of the file and populates the questions and all available attributes on the GUI. It also initializes the different data structures.
- **Create New Type**: It reads the short description and the possible answer choices from the GUI and stores all the answer choices in a variable such that each choice is separated by a "/". This information is then stored in the data structure and is persisted in the file using the Constants class.

Figure 11 is the class diagram representation of the FormQuestionnaire class.

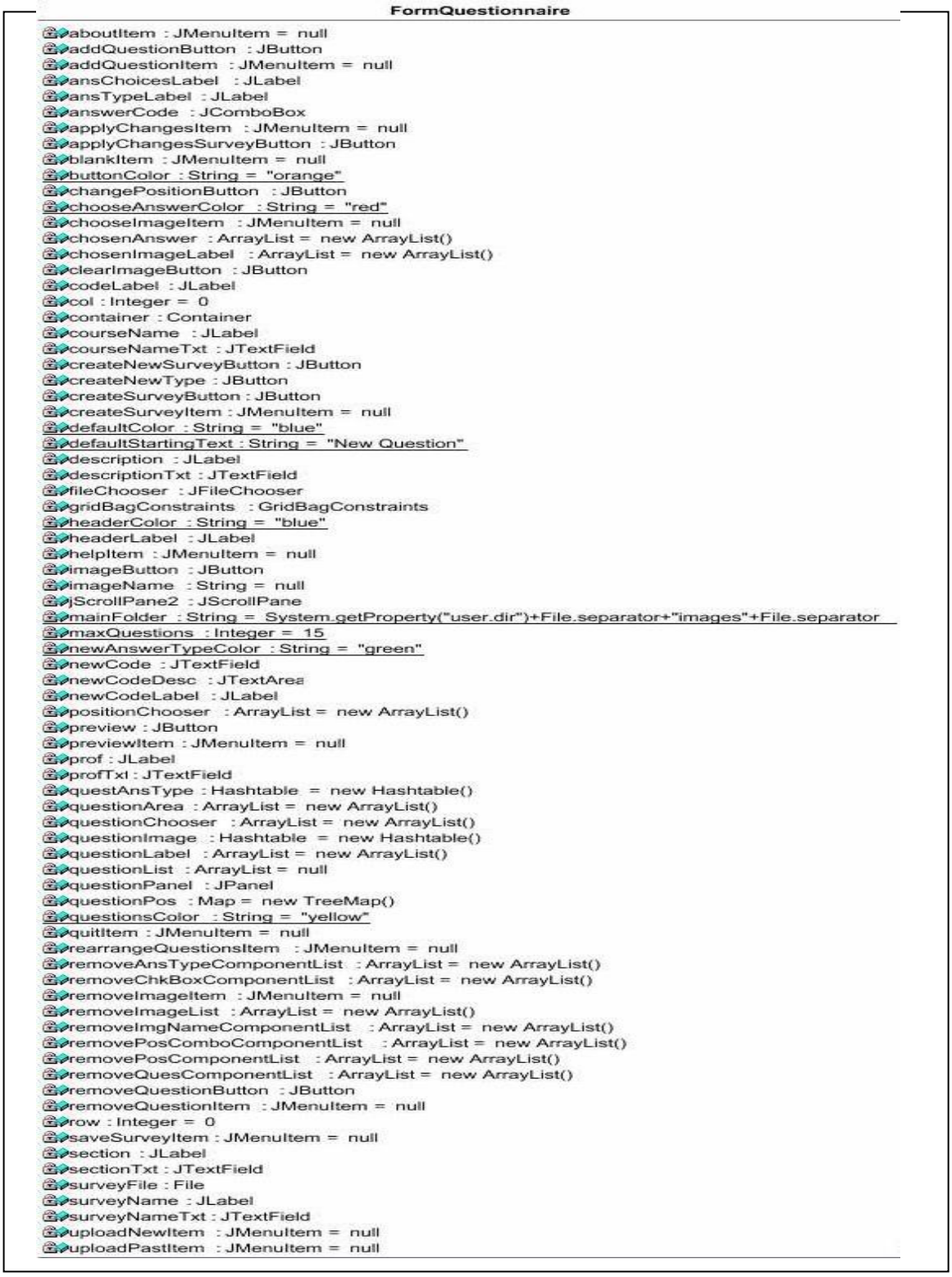


Figure 11 - FormQuestionnaire Class diagram. a) Data Members

Figure 11 - FormQuestionnaire Class diagram. b) Method Members

5. IMPLEMENTATION

An important component of implementation constituted the design of the survey text file format. We had to come up with a format that facilitated easy parsing and modification. The agreed-upon format for the survey text file (MC formatted file) consists of the header section and the questions section.

The header section consists of the following fields:

Survey Name [tab] Course Name [tab] Course Description [tab] Professor [tab] Section

The questions section consists of the following fields for every question:

Answer Type [tab] Question [tab] Answer choices [tab] [IMG] Image Name

The answer type field is used to indicate if the question is a multiple choice, text or an essay type question. A question for which the answer type has not been selected has a value of -1 in the answer type field. The [IMG] tag is used to indicate that the following string represents the image name.

The implementation of the various classes is discussed in the following sections.

5.1. Constants Class:

This class loads all the application properties into a data structure that is shared by the entire application. It also loads all the standard answer types as well as the answer types added by the user by using the Properties.load() method. Any time the user adds a new answer type; this class inserts the new answer type into the constants.properties file and at the same time, updates the data structure.

5.2. ArrayWorker Class:

As mentioned in the previous section, this class is responsible for parsing a list of strings and returns a list of question objects created. Each object thus created will reflect the information last saved by the user. As mentioned in the previous section, “tabs” in the file separate each attribute of the question and each line in the text file represents a question. The “parseAndGenerateQuestions()” method accepts a list of strings, each string representing a question (read from the MC formatted file using the “FileWorker”). The method tokenizes each string and sets corresponding attributes in the question object.

All such question objects are then returned to the caller as a list of question objects. If a question has an image associated with it, the name of the image is stored at the end of the string, as per the MC file format, beginning with the "[IMG]" tag.

5.3. FileWorker Class:

As discussed in Section 4, the FileWorker class reads data from a given file and returns a list of strings, each representing lines of text in the file. If the file being read from is an MC formatted file, then each line represents a separate question along with its attributes, delimited by "tabs". The first line in this file represents the survey header information, which is also separated by tabs.

This class is also used to create the MC formatted file using the "generateIntermediateFile()" method. This method expects two arguments - fully formed survey header information and a list of question objects. The header contains the survey name, at the very least, since it is required in order to save the survey. If the information for the other fields is missing, each such field will contain a hyphen as a placeholder and will be replaced with a value specified by the user at a later point of time.

5.4. SurveyWorker Class:

This class is implemented to generate the survey files that will be deployed on the target server. When the survey files are generated, three files in the format that is consistent with the current system are generated along with any images that were attached to each question. The three files include the text file, HTML file and the JSP file. These files are generated under a folder created for each survey and all the surveys generated by the user are grouped under a common folder called "survey". The name for the common folder can be changed in the property file without having to change any code. In the case of preview, only the HTML file along with any images that were attached to each question is created. As in the case of final survey files generation process, a separate folder is created for each survey when the preview files are generated and all such preview folders are grouped under a common folder called "preview", which can again be changed by changing the property file. The MC formatted file is saved under

a common folder “tmp” for all the surveys. Again, the name for the common folder can be changed in the property file without having to change any code. The format of the file name is [SurveyName+”TmpSurveyFile”+ext]. The “ext” is the extension of the MC formatted file, which is read from the Constants class. The file extension can be changed by changing the value in the properties file.

The JSP and the HTML files are generated using template files. By changing the generic content in these files, all the future generated files will change accordingly. These files contain placeholders for survey specific data along with generic information that is shared among all surveys. The JSP template as shown in Figure 13 includes the HTML file that is generated as part of the survey generation process. The HTML file that is generated for this purpose contains only the information (attributes) about each question and does not make use of the HTML template. The HTML template as shown in Figure 12 is used only when the preview feature is used. This contains all the information contained in the JSP template (without the JSP tags) so that the user can have the same experience as the final survey file generated.

```
<h1>Survey ###survey_number### Assessment in ###course_number###</h1></center>
…
###html_body###

<input type="submit" value="Submit survey"> ( Make sure you are sure of your
replies before submitting. You should only submit one survey)
```

Figure 12 - HTML template

```
<% String surveyName = "###survey_number###"; %>
…
<h1>
Survey  <%= surveyName %> Assessment in ###course_number### </h1></center>
…
<jsp:include page="<%= surveyName+".html" %>"  />
…
```

Figure 13 - JSP template

When these files are generated specifically for a survey, the placeholders are replaced with the appropriate values specified by the survey creator. The "generateHTMLFileData()" method generates the HTML content to be written to the file. When the question text is read, the string is scanned for the special character that was introduced during save or generation of the MC formatted file. If the special character is found, then it is replaced with the HTML equivalent of the newline character (
).

This class provides methods to copy the images from the users file system into survey-specific folders generated using the FileChannel's "transferFrom(srcChannel, 0, srcChannel.size())" method. When the survey files are generated, it also generates a zip file containing all the survey related files by using the ZipOutputStream class.

5.5. DataProcessor Class:

This class acts as a mediator between the FormQuestionnaire class, where all the user interactions are captured and handled, and the worker classes. In some cases, this class accepts the request and returns the result from the worker classes. The "readDataFile()" (to read the data from the file) method and "parseAndGenerateQuestions()" (to parse and generate a list of question objects using lines contained in a list) method pass on the request to FileWorker class and ArrayWorker class respectively and do no additional processing. In other cases, this class does some pre-processing before forwarding the request to the worker class. The "generateTempFile()" method accepts the MC formatted file name, the header information and a list containing the information about all the questions. This method creates a new directory, pre-processes the filename to contain the correct path, parses and generates a list of question objects and then uses the FileWorker class to generate the MC formatted file. In other cases, it interacts with more than one worker class to respond to the service requestor. The "getDataFromTmpFile()" method reads the data from the MC formatted file using the FileWorker's "readDataFile()" method, processes the result and then uses the ArrayWorker class to generate the list of question objects. This method returns a list of question objects followed by the header information.

5.6. LaunchBrowser Class:

This class is used to display the HTML file generated when the preview functionality is used. It checks if the user's operating system is windows or non-windows. If it is windows, it calls the runtime "dll" to open the Internet browser and displays the generated HTML file within it. This uses the Runtime class to execute an external process from the application. If it is non-windows, it uses the internal ISDK browser, which is a non-editable JEditorPane. This JEditorPane is capable of displaying HTML content.

5.7. FormQuestionnaire Class:

This class is the main class that captures all user interactions and provides a means to display the result of the user's action.

A user can rearrange the order of the questions as they should appear in the final survey. There are two ways that the user can achieve this. If more than one question needs to be rearranged, when the user clicks on the "Rearrange Questions" button after setting the new question numbers, the application first reads the data on the screen and creates question objects for each question on the screen and sets all the question's attributes. As soon as the question objects are created, it is inserted into a list at the position that the user had specified for it on the screen. At the end of this process, the list is populated with question objects, each set in its appropriate position equivalent to the position on the screen. The questions are then read from this list in a sequential order and set on the screen, in the same order. At the same time, all the remaining attributes of the question are also set using the information stored in the question object. If there is only one question whose position needs to be changed, the user can choose the question and then select the new position. When the state of the position changes for the question and if the question was chosen**,** the application checks to see if the new position is after the current position or before the current position. If the new position is after the current position, then the positions for all the questions between the current position and the new position are decremented by one. If the new question is before the current position, the positions for all the questions between the current position and the new position are incremented by one. Next, the same procedure explained above for the case where

multiple questions are rearranged is carried out. The end result of these steps is the placement of the questions in a new position along with its attributes.

There are two cases to handle when dealing with adding new questions to the survey. The new question can be added at the end or at any specific position. Adding a new question at the end is straightforward - a row (with the question chooser checkbox, position chooser drop-down, question area, answer type label area and image name label) is created for the new question and displayed at the end. Inserting a new question at any other position is an extension to the above step. The new question is first added at the end and then the application detects before which question the new question needs to be inserted. Then, the procedure described for the "rearrange questions" is carried out to have the effect of inserting a question before a specified position.

When a question needs to be deleted, the user selects the question and clicks on the "Remove Question" button. The action listener for this button is fired and the procedure implemented for this action is executed. In this case, the question text for the question that needs to be deleted is first cleared out and then the information that is remaining on the screen is captured in a list of question objects. During this process, a question object for the question that does not have the question text is not created and hence, the object will not exist for the question in the final list. Next, using the list just created, all the question attributes are re-displayed on the screen. When this process is executed, all the attributes for the question that was deleted are removed from the screen and all the questions under it are displayed one position above. This, however, leaves the last question duplicated in two positions (last and the one before last). Hence, care was taken to clear the replicated question.

When the user enters a new answer type and the "Create New Type" is clicked, the application reads the short description and the possible answer choices entered in different lines. The answer choices are then combined together in a string with each option separated by a "/" (which is the delimiter used to separate the different answer choices). Both the short description and the answer choices are then passed on to

Constant class's "addConstant()" method to be saved in the static data structure and also to update the constants.properties file.

The relationship between the question and the answer type is stored in memory and not written to the MC formatted file until the user wishes to save the survey. When a question is chosen and the answer type is selected for the question, the new relationship is updated in the data structure when the "Apply Changes" is clicked and the action listener for this button is executed.

The procedure implemented for the "Save Survey" feature is triggered when the user clicks on the "Save Survey" menu item. All the information on the screen that is not saved in the data structure is first saved. This step is needed to capture the last change and to have the same effect as clicking on the "Apply Changes" and then "Save Survey", since the user can decide to save the survey before clicking on "Apply Changes". Next, a list of strings, each representing the question attributes, is generated. The format of the string is a mirror of the MC formatted file content. Since each question needs to exist on a single line, all newline characters read from the question area are converted to a special character before writing to the MC formatted file. When the saved survey is displayed again, these special characters are again reverted to the newline characters so that they are displayed on separate lines. Next, all the header information is captured and stored in a string in the format compatible with the MC formatted file. If there are any fields missing in the header, a "-" placeholder is used in place. All the information captured is then passed on to the DataProcessor's "generateTempFile()" method in order to create the MC formatted file.

Starting a new survey using the File → New Survey → Blank Survey option is equivalent to starting up the application. This disposes the current frame and brings up a new frame with the default one question and populates the question as the initial startup does.

Uploading survey questions from the file instead of manually entering each question on the GUI using the File → New Survey → Upload Survey Questions option first collects any header information that was entered by the user, disposes the current frame and then reads the data from the chosen input file using the DataProcessor's "readDataFile()" method. Next, the header information that was collected is populated and then all the questions that were read from the input file are uploaded on the screen.

When an MC formatted file is uploaded using the File → New Survey → Upload Past Survey option in order to continue working on a previously saved survey, the DataProcessor's "getDataFromTmpFile()" method is used to create a list of questions and also to get the header information from the saved file. First the current frame is disposed and all the data structures are re-initialized to mimic the state at which the user stopped working on the survey. Next, the questions are re-populated on the screen using the question objects retrieved from the DataProcessor. The process of uploading a past survey is completed by populating the header and all available question attributes on the screen.

When the user wishes to preview the survey that was created so far, internally the application reads and populates the data structures from the screen that were not already saved by the user. Next, a list of question attributes string is generated and passed on to DataProcessor's "parseAndGenerateQuestions()" method to generate a list of question objects. Next, this information is passed on to DataProcessor's "generateHTMLFileData()" method to generate the HTML body. Then, the header information is gathered and a hash table is created with all these values. This data structure (as shown in Figure 14) and the name of the template is then passed on to DataProcessor's "generateSurveyHTMLFile()" method to generate the complete HTML content with all the values filled in the template.

```
Hashtable surveyFillinValues = new Hashtable();
surveyFillinValues.put("###survey_number###",surveyNumber);
if ( courseNumber != null && courseNumber.trim().equals("") )
        courseNumber = surveyNumber;
surveyFillinValues.put("###course_number###",courseNumber);
surveyFillinValues.put("###course_desc###",courseDesc);
surveyFillinValues.put("###section_code###",sectionCode);
surveyFillinValues.put("###html_body###",htmlContent);
```

Figure 14 – Data structure to hold the values to be filled in the template

As the final two steps, all the images associated with this survey are also copied into the directory created exclusively for this survey and then the HTML is created in the directory using the DataProcessor's "generateSurveyHTMLFile()" method. The generated HTML is launched in the browser using LaunchBrowser's "previewURL()" method.

The final survey files can be generated only after all the questions are fully formed. Generating the final survey files is much like the preview but for some differences. All the steps outlined in the preview process are carried out. In this case, the entire HTML file content is not generated. Only the main HTML body is generated with the HTML code for displaying the questions (along with their attributes). Once this is obtained, a separate directory is created for the survey and the HTML file is created containing only the HTML body (without the information in the HTML template). Next, the survey specific data structure is populated as in the preview case (values to be plugged into the JSP template) and passed on to DataProcessor's "generateSurveyJSPFile()" method to generate the JSP file in the survey specific directory. Along with this file, all the images associated with the survey are copied to this directory. In addition to this, a text file in the format compatible with the current survey system is generated using the DataProcessor's "generateSurveyTextFile()" method in the same directory. When all the necessary files are generated, a zip file containing these files is created so that it can be deployed on the target server.

6. Usage of the Application

When the application is executed, a splash screen similar to the one in Figure 15 appears as follows:

Figure 15 - Welcome Splash Screen

In order to view the main application, the user can click anywhere on the splash screen and the main application window appears as shown in Figure 16.

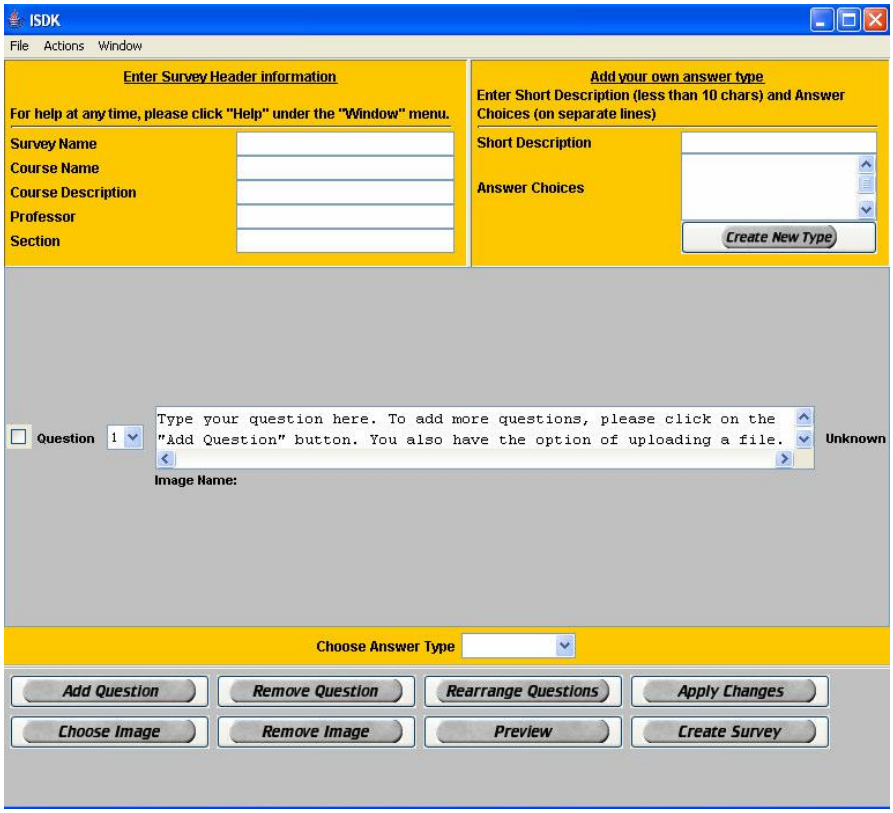


Figure 16 – Main application window

The Header portion at the top allows the user to enter information such as the Survey Name, Course Name, Course Description, Professor Name and Section Number. This information will be used in the JSP template used to create the JSP file. The Survey Name is a mandatory field without which the user cannot save or create the survey.

The screen initially displays a single question slot containing the message "Type your question here". The user can either upload a text file containing the questions or can enter questions manually. To choose either option, he has to click on the “File” menu at the top which displays options such as Start Survey, Save Survey and Quit as shown in Figure 17.

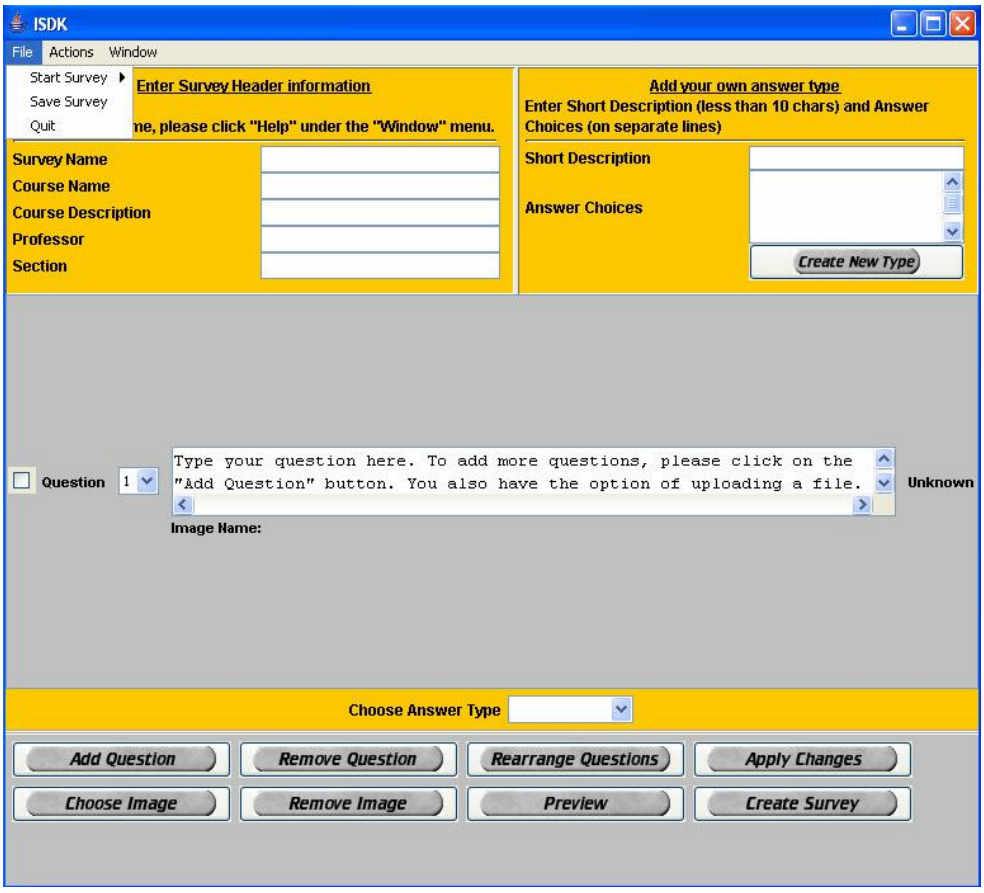


Figure 17 – Menu option listing

When the user places the arrow on the Start Survey option, a sub-menu opens showing options such as Blank Survey, Upload New Survey and Upload Past Survey as shown in Figure 18.

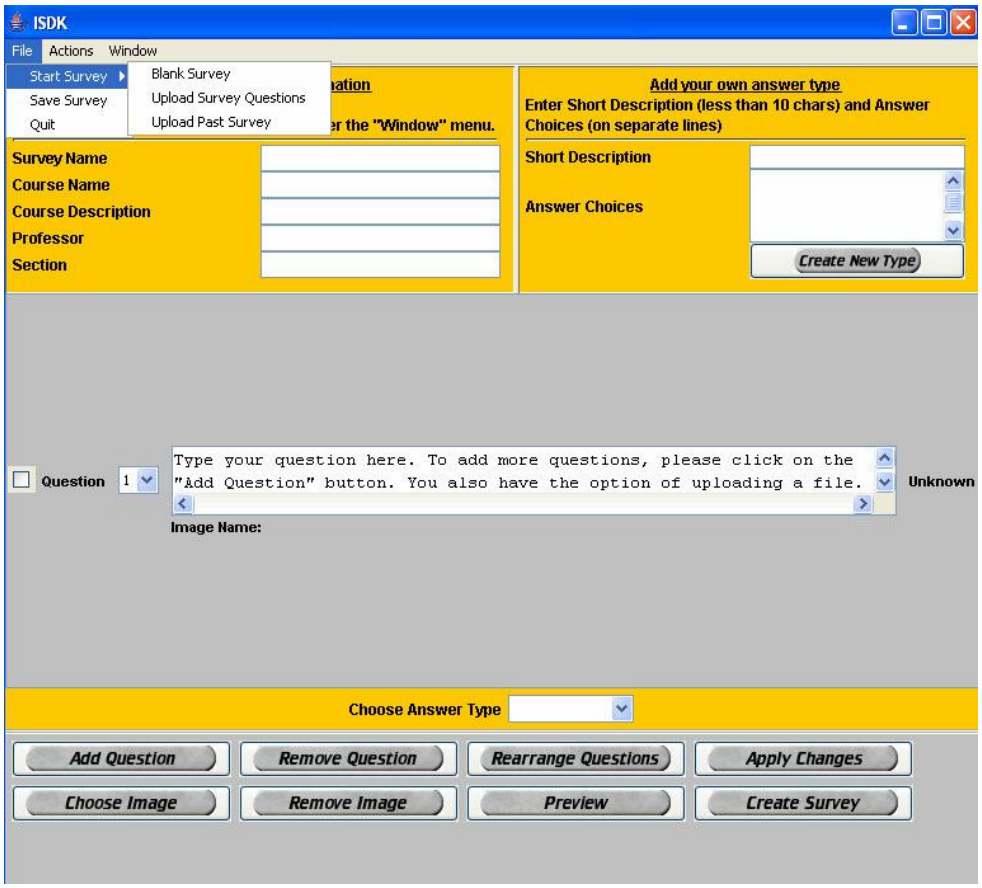


Figure 18 – Sub menu option listing

Clicking on the “Blank Survey” menu option allows the user to add questions manually. The screen initially displays a single question slot containing the message "Type your question here". The drop-down menu for the question numbers contains one at this point. Once the user has typed a question in this slot, he can add any more questions by clicking on the "Add question" button. Every time a question is added, the number in the drop-down menu is incremented automatically. The user can either add a question at the end or at any specific position. In order to add a question at the end, the user has to just click on "Add question" button and an empty slot for the question will be added at the end as shown in Figure 19.

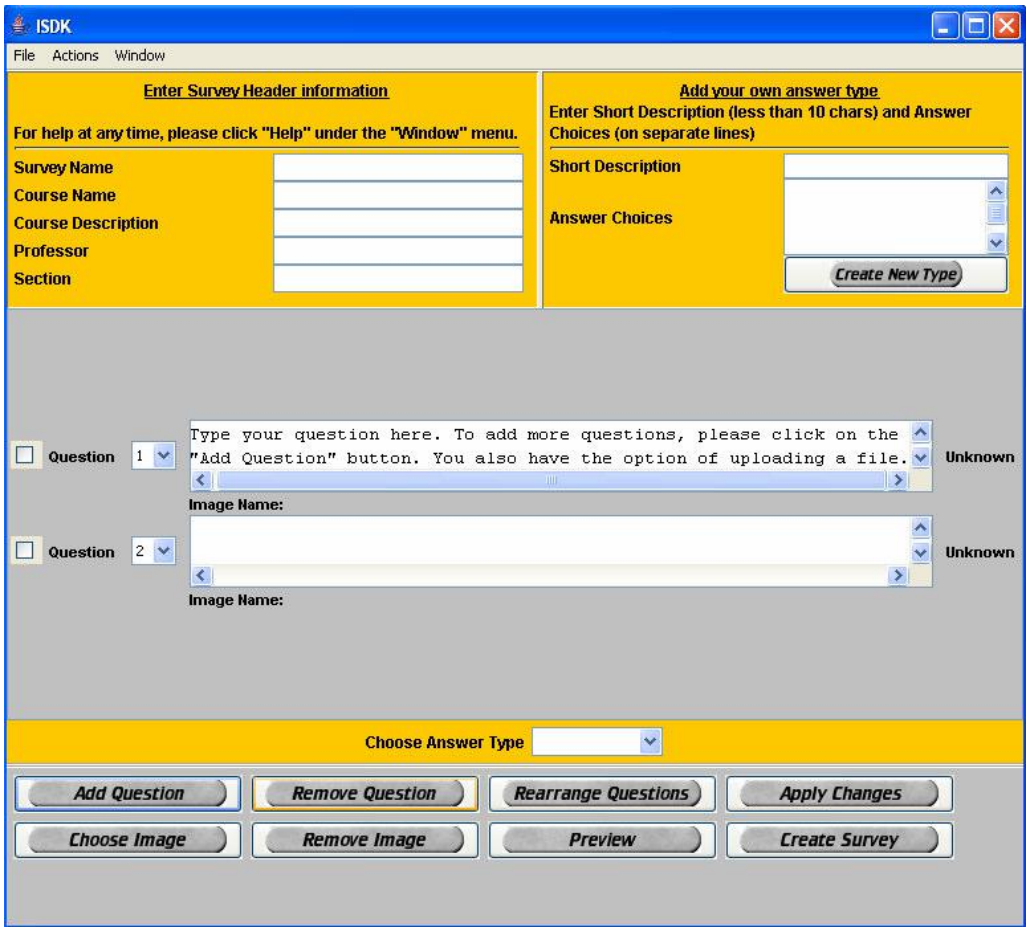


Figure 19 – Demonstrating adding a question at the end

If the user wants to add a question at any specific position, the user has to click on the question before which he wants to add the new question and then click on the "Add question" button. The new question is inserted before the question that was clicked. For example, if the user wants to add a new question before the second question, he has to select the second question and then clicking on the “Add Question” button inserts a new question at the second position as shown in Figure 20.

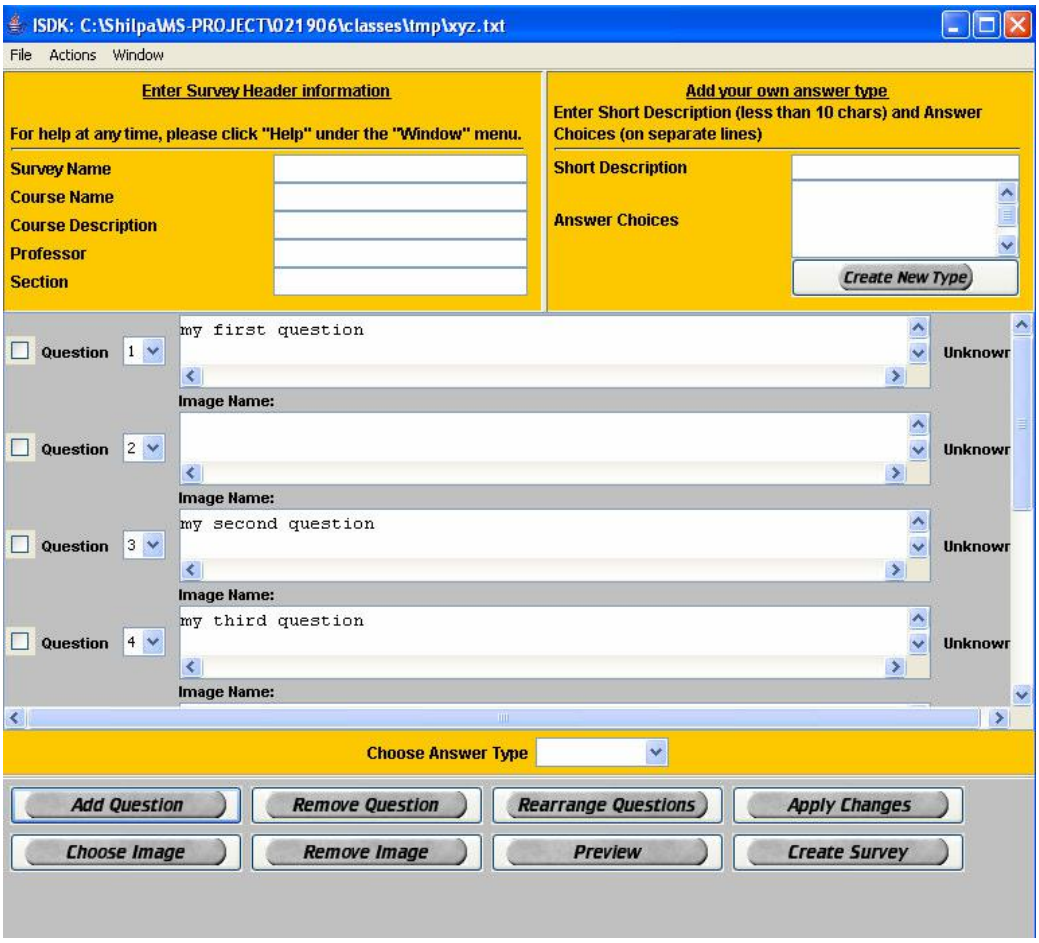


Figure 20 – Demonstrating adding a question at a specific position

The "Upload New Survey" menu option allows the user to browse through the system and choose a text file containing the questions that can be uploaded. When the user clicks on this option, the "Choose Input Survey file" dialog box opens as shown in Figure 21.

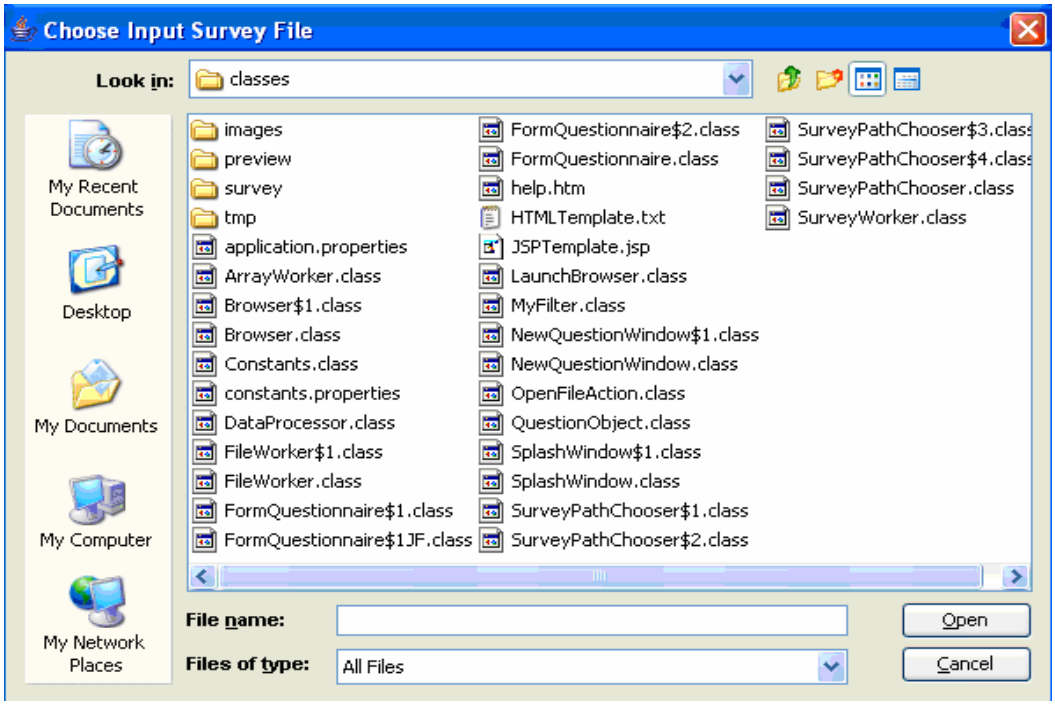


Figure 21 – Choose input survey file dialog box

Once the user selects a text file and clicks on the Open button, the questions in the text file are uploaded onto the screen as shown in Figure 22.

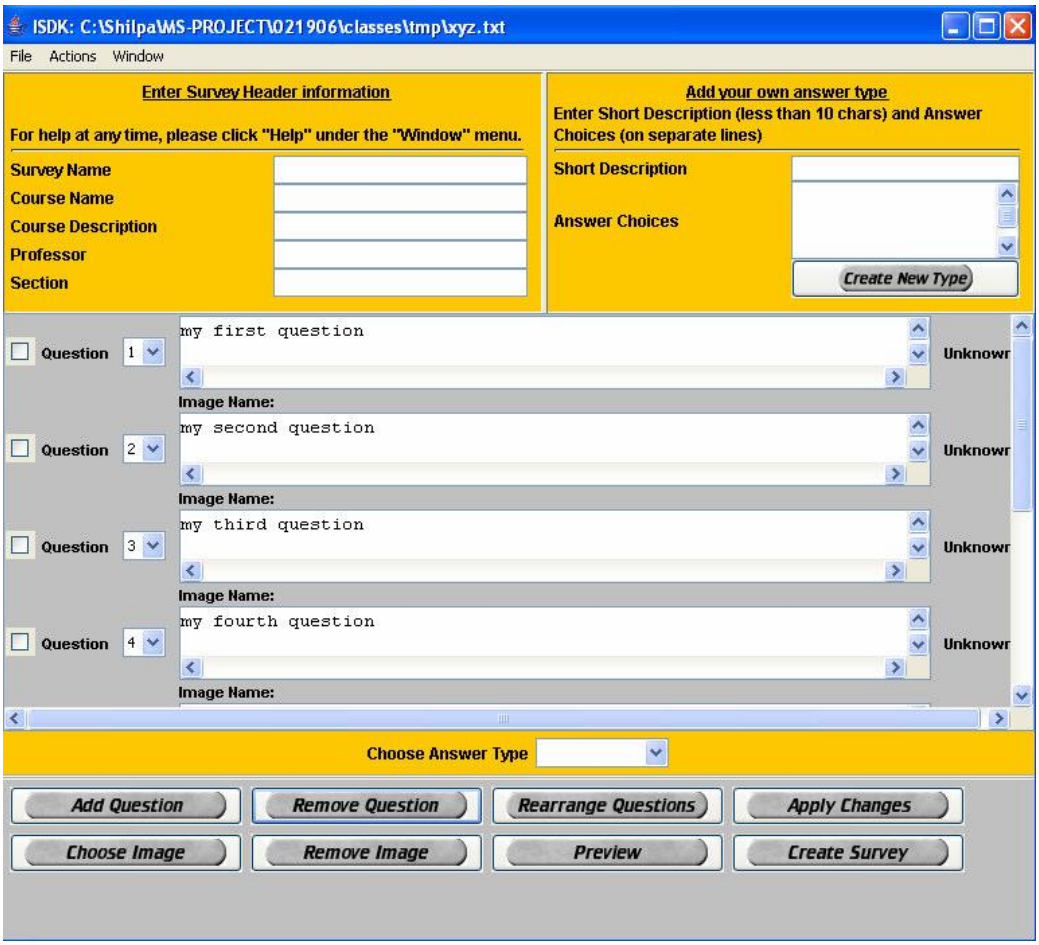


Figure 22 – Demonstration of uploading a text file

The questions will appear in the same order as the text file or the order in which the user entered the questions and every question is assigned a question number. Every question number has a drop-down menu that lets the user change the order of the questions if he wishes to do so. There are two ways that the user can choose to rearrange the questions. The user can check the box next to the question and choose a new position for the question. For example, if he wishes to make the first question the third question, he has to check the box next to the first question and select number three in the drop-down menu as shown in Figure 23.

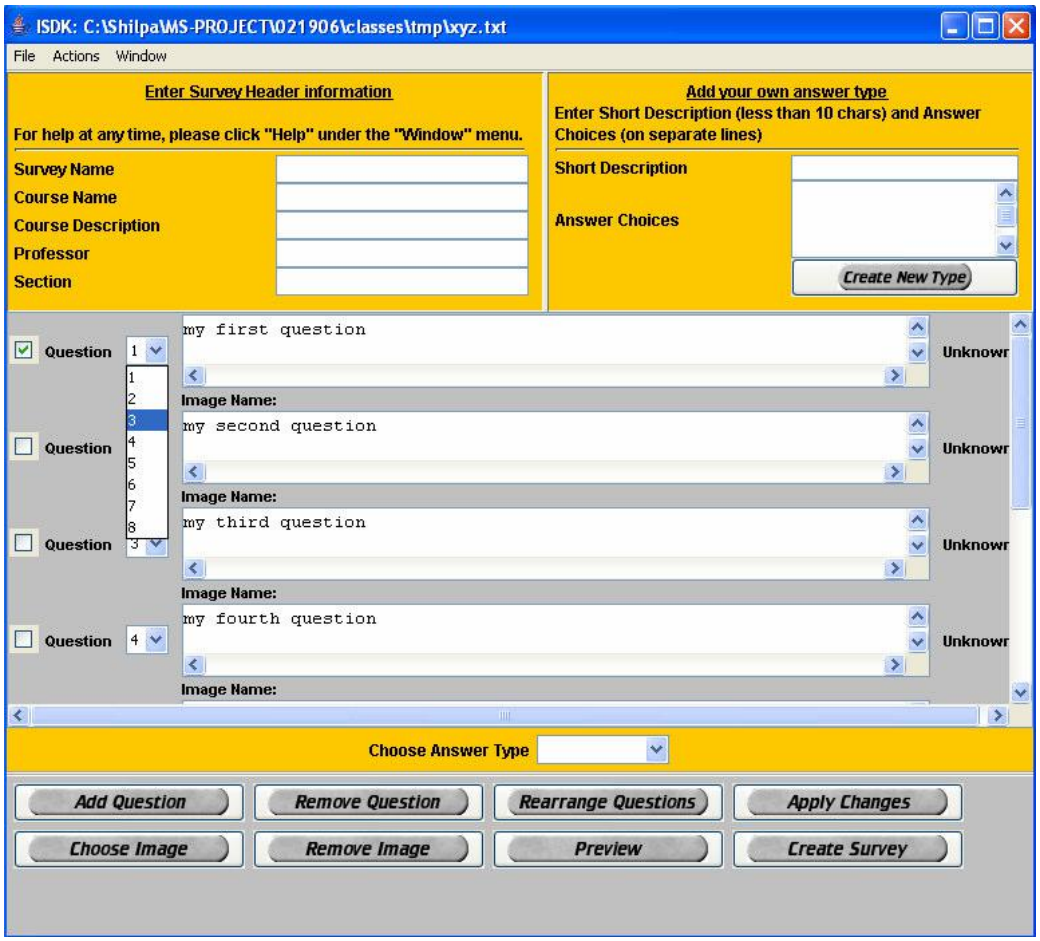


Figure 23 – Demonstration of rearranging a single question

When this action is performed, the questions are automatically rearranged as shown in Figure 24.

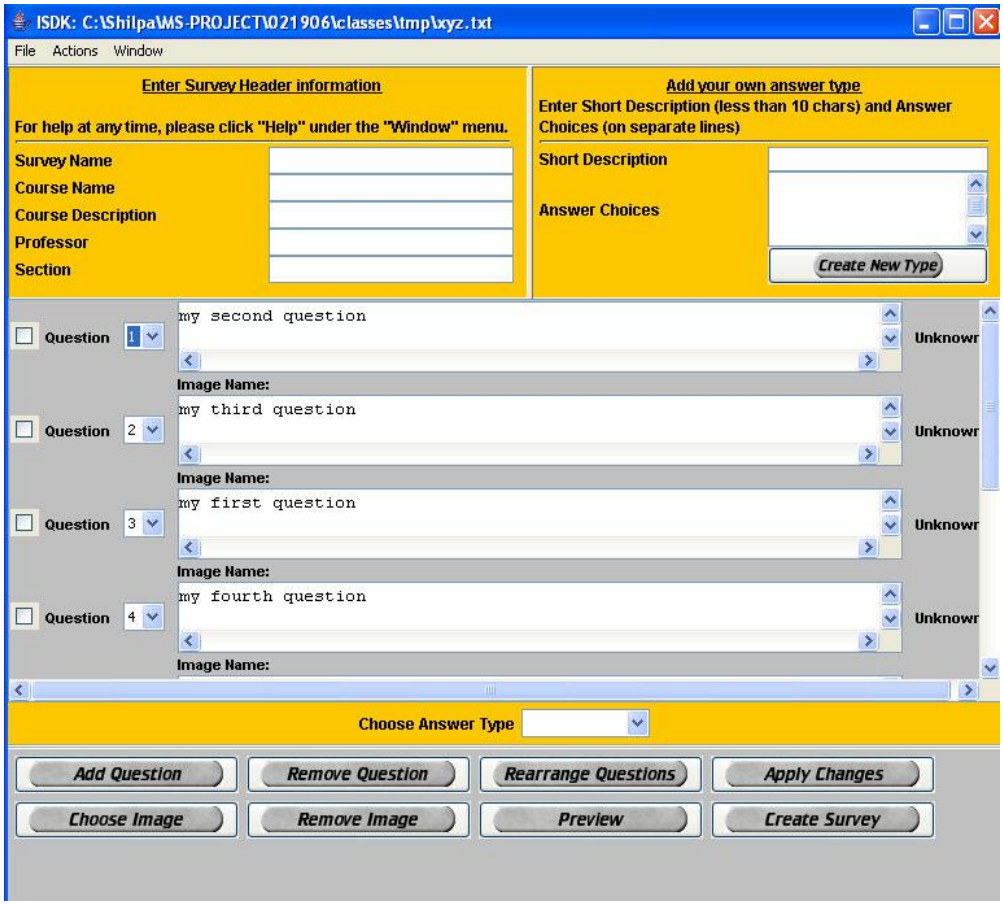


Figure 24 – Rearrangement of questions on the screen

If more than one question needs to be rearranged, the user can choose the new positions for all the questions that need to be changed. Once this is completed, the user can click on the "Rearrange Questions" button. For example, if the user wants to reverse the order of the first four questions, he has to select the new positions for the questions in the drop-down menu as shown in Figure 25.

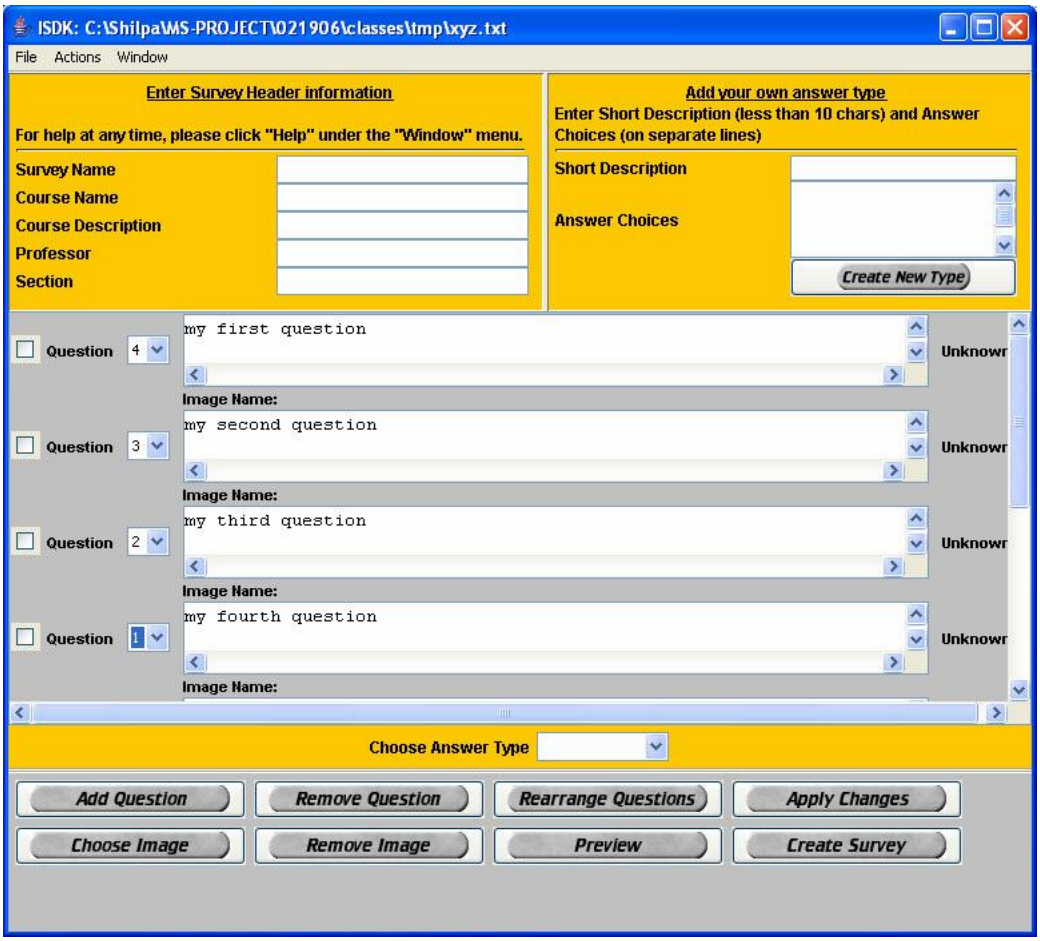


Figure 25 – Demonstration of rearranging multiple questions

When he clicks on the "Rearrange questions" button, the changes are reflected immediately and the questions will appear in the updated order as shown in Figure 26.

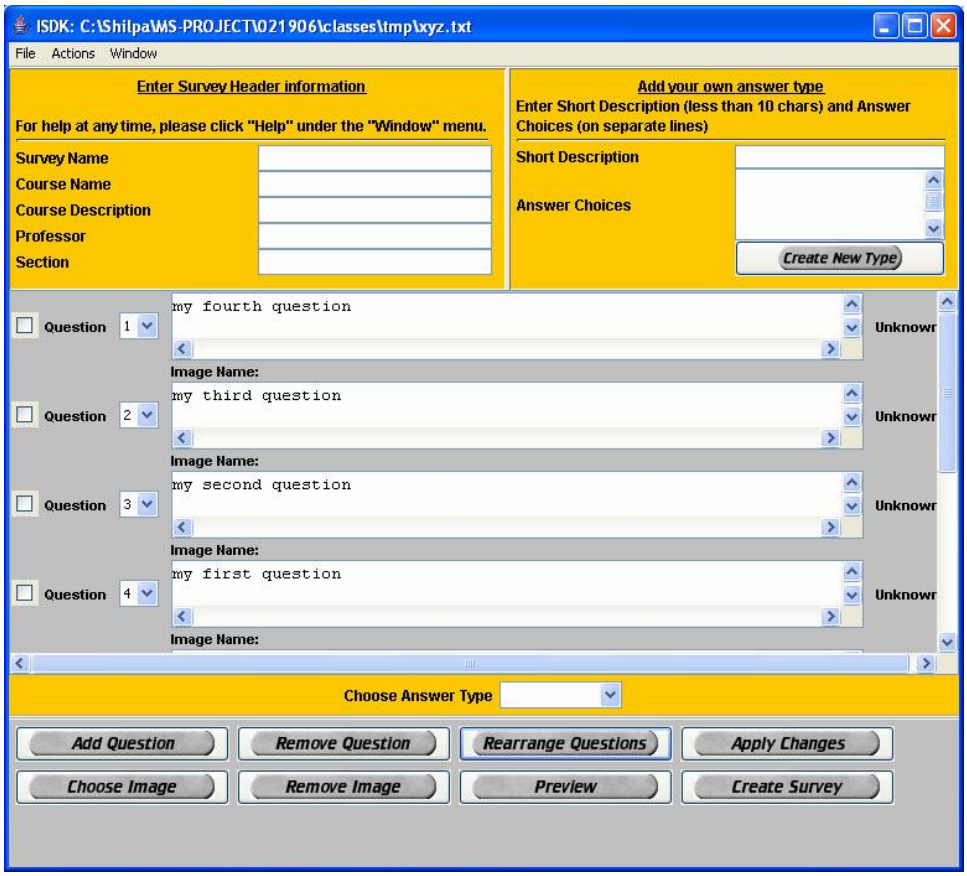


Figure 26 – Rearrangement of multiple questions at once

The ***"Remove question"*** feature allows the user to delete either a single question or multiple questions. The user has to select the questions that he wants to delete and click on the "Remove question" button. For example, if he wishes to delete the second and third question, he has to check the box next to these two questions and clicking on the "Remove Question" button will delete these two questions as shown in Figure 27.

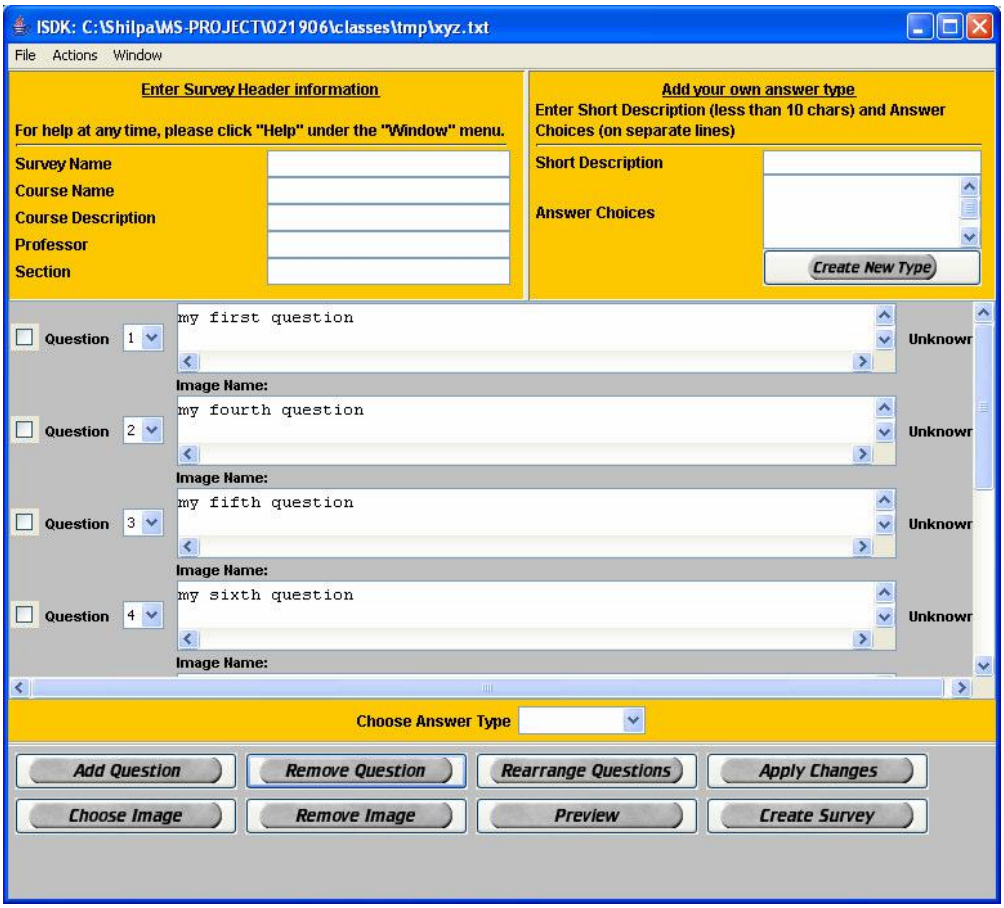


Figure 27 – Demonstration of deleting questions

While selecting answer choices for each question, the user can either select answer choices from a list of standard answer types or can provide his own. The answer choices are displayed next to the questions on the GUI. Initially since the answer choices have not yet been selected for the questions, they show up as "Unknown". The drop-down menu next to "Choose Answer Type" provides a list of the standard answer types as shown in Figure 28.

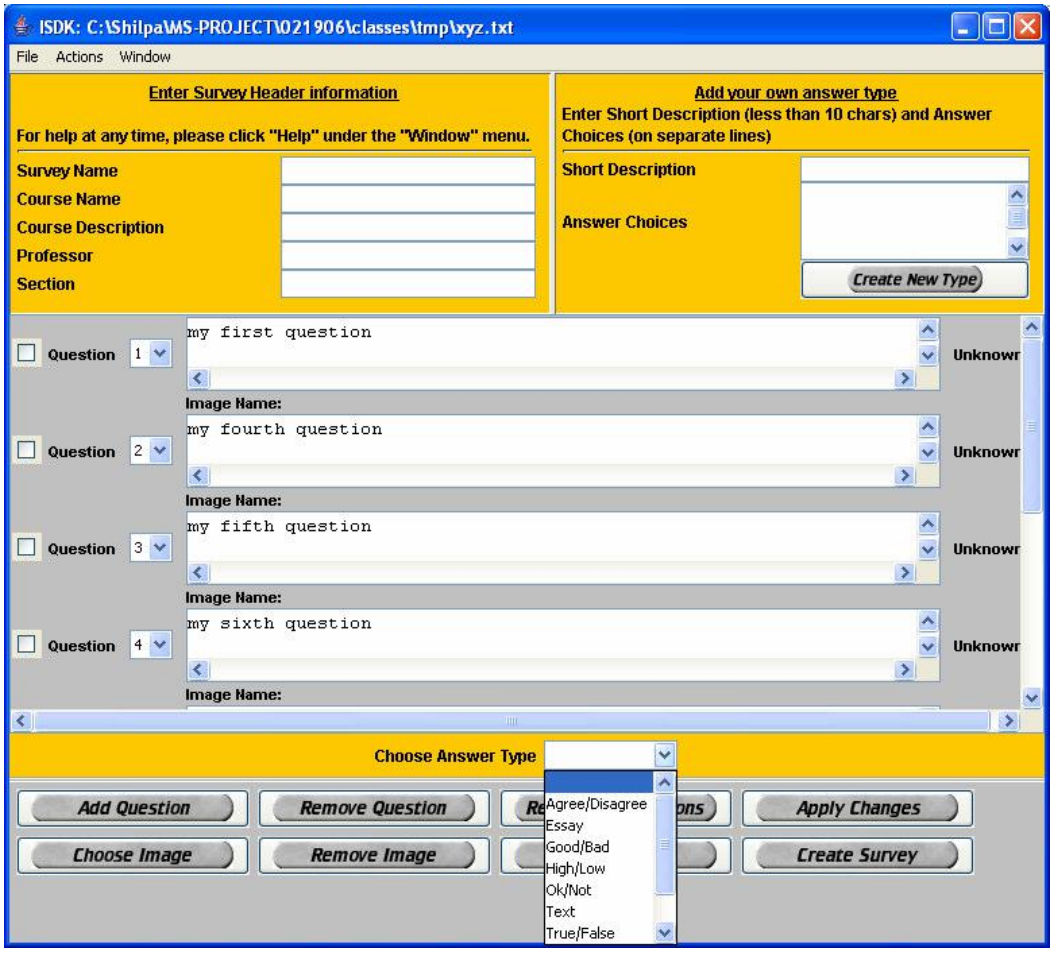


Figure 28 – Listing of standard answer choices

The user can select a particular answer type for a single question or for multiple questions at once. The user has to select the questions for which he wants a particular answer type and select the answer type from the drop-down menu. For example, if he wants to select True/False for the first and third question, he has to check the boxes next to these questions and select True/False from the drop-down menu as shown in Figure 29.

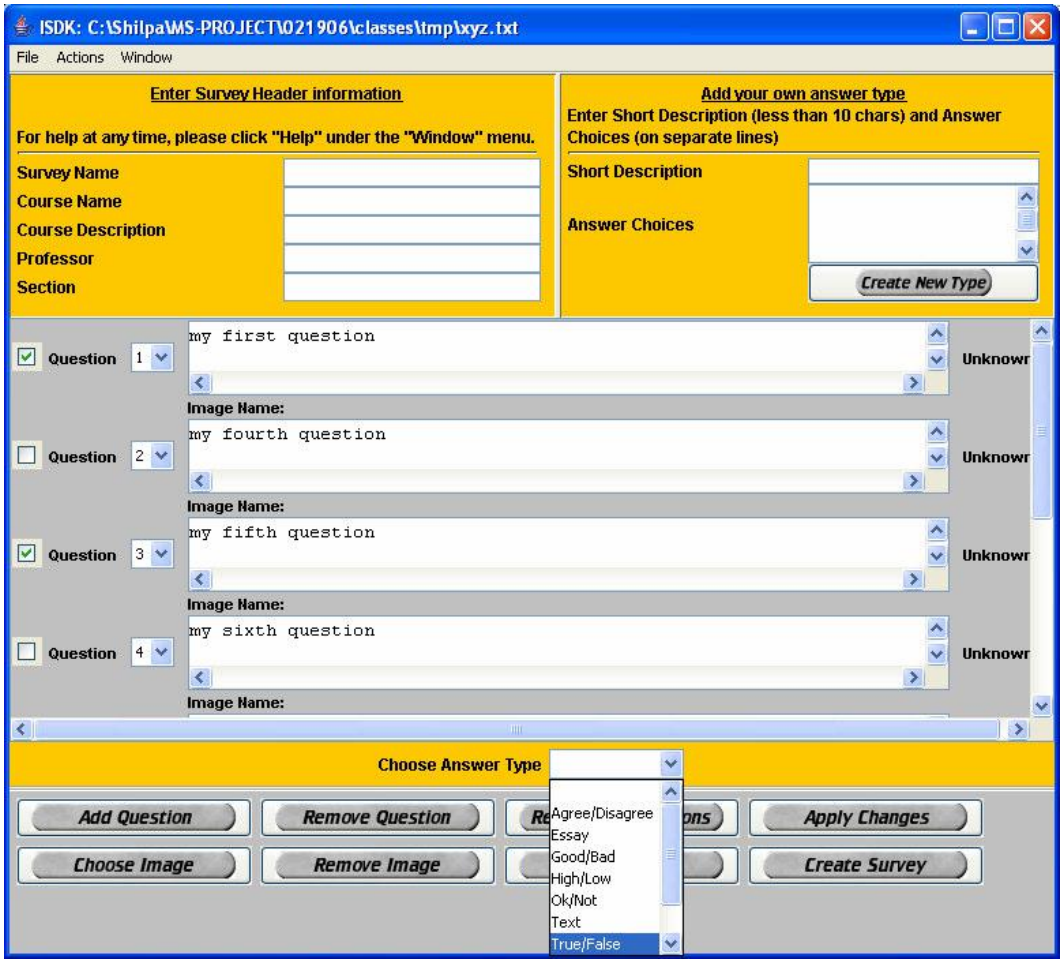


Figure 29 – Demonstration of choosing a standard answer type

Once he has selected the answer type, clicking on "Apply changes" button will display the answer choices next to the questions as shown in Figure 30.

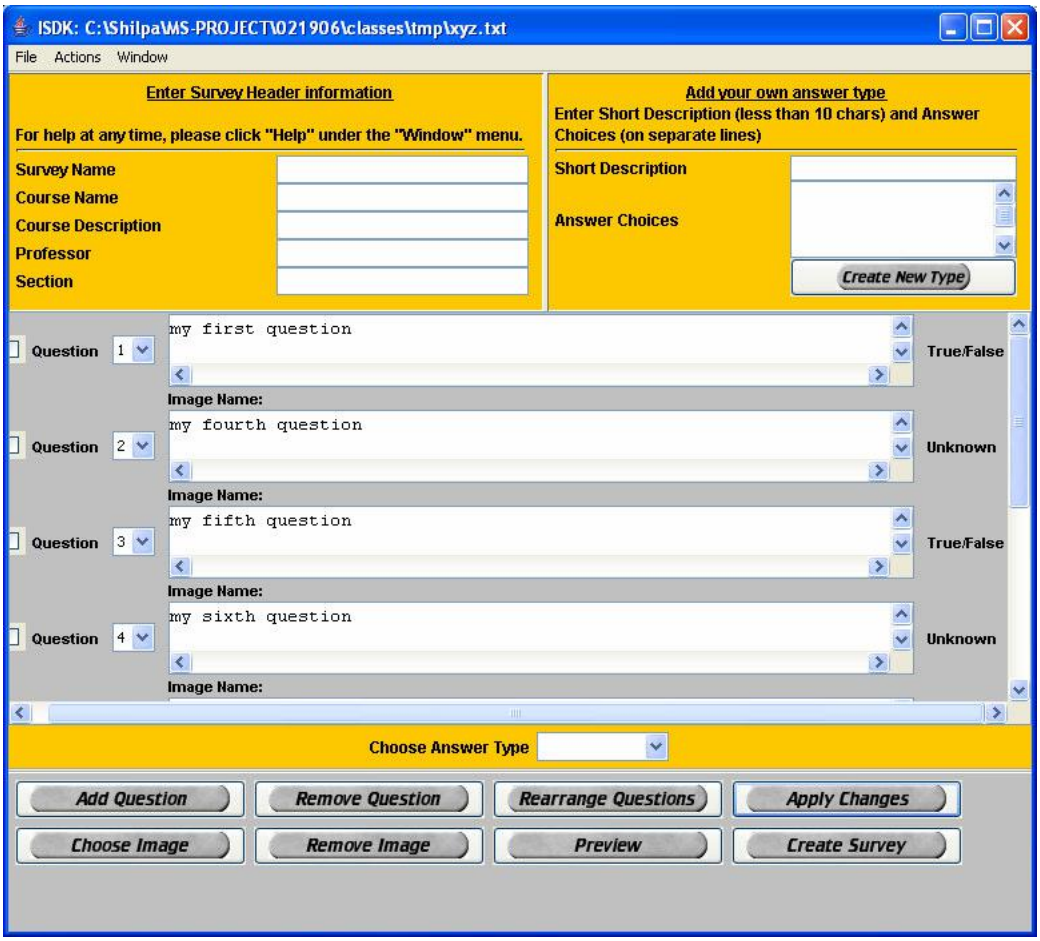


Figure 30 – Displaying answer choices next to the questions

If the above answer choices are not appropriate, the user can add his own answer type. To do this, he has to first enter a short description that he wishes to use to represent the corresponding answer choices. He then has to provide the actual answer choices one on each line in the field called "Answer Choices". For example, if the user wants to use answer choices Excellent / Good / Bad / Poor, he can select any description such as Good / Bad and provide the choices Excellent / Good / Bad / Poor, each on a separate line in the "Answer Choices" field as shown in Figure 31.

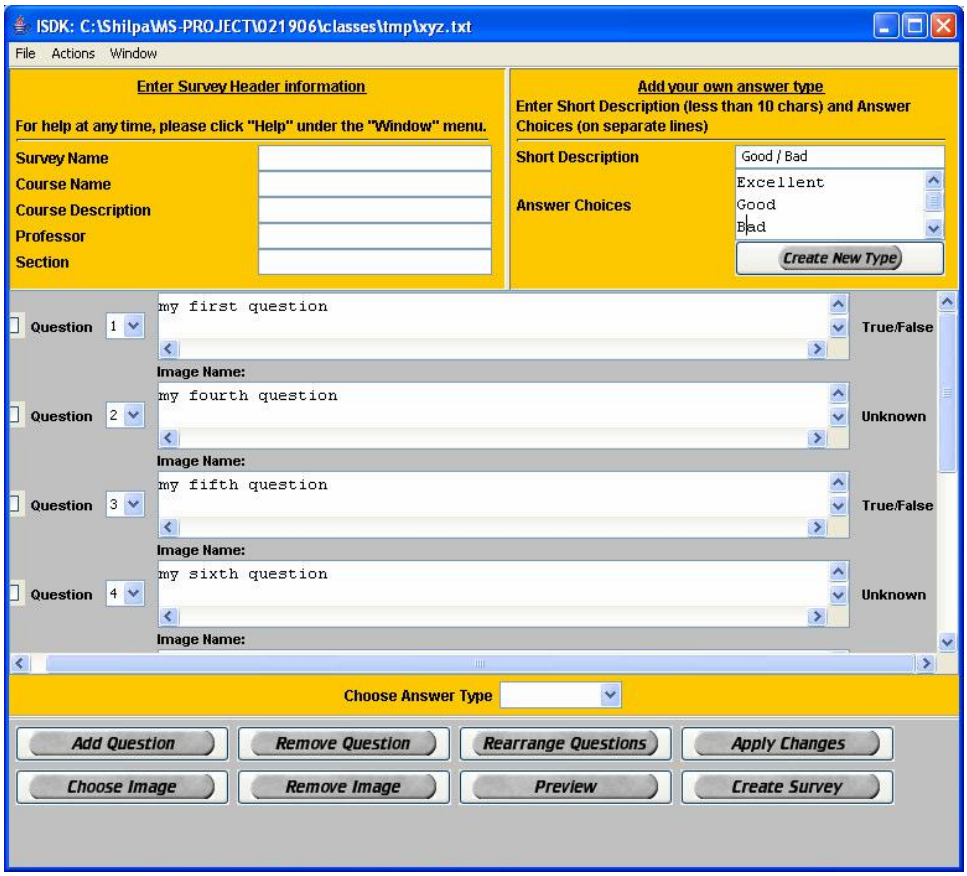


Figure 31 – Demonstration of creating new answer types

Once he has finished entering all the answer choices, he can click on the “Create New Type” button which adds this new type to the list of existing standard types. The advantage of doing this is that the next time he wants to use the same answer choices, it is available rather than having to re-create it each time. Since the drop-down menu contains the newly created answer type, he can assign these new answer choices the same way as the standard answer choices. If he wishes to change the answer type for any question, he can change it by selecting the question and choosing a different answer type from the drop-down menu and finally clicking the “Apply Changes” button.

The ***“Choose Image”*** feature allows the user to append an image to any question. The user can attach an image to a single question or to multiple questions at once. The user has to select the questions for which he wants to append a particular image and once he clicks on the "Choose Image" button, a dialog box opens as shown in Figure 32.

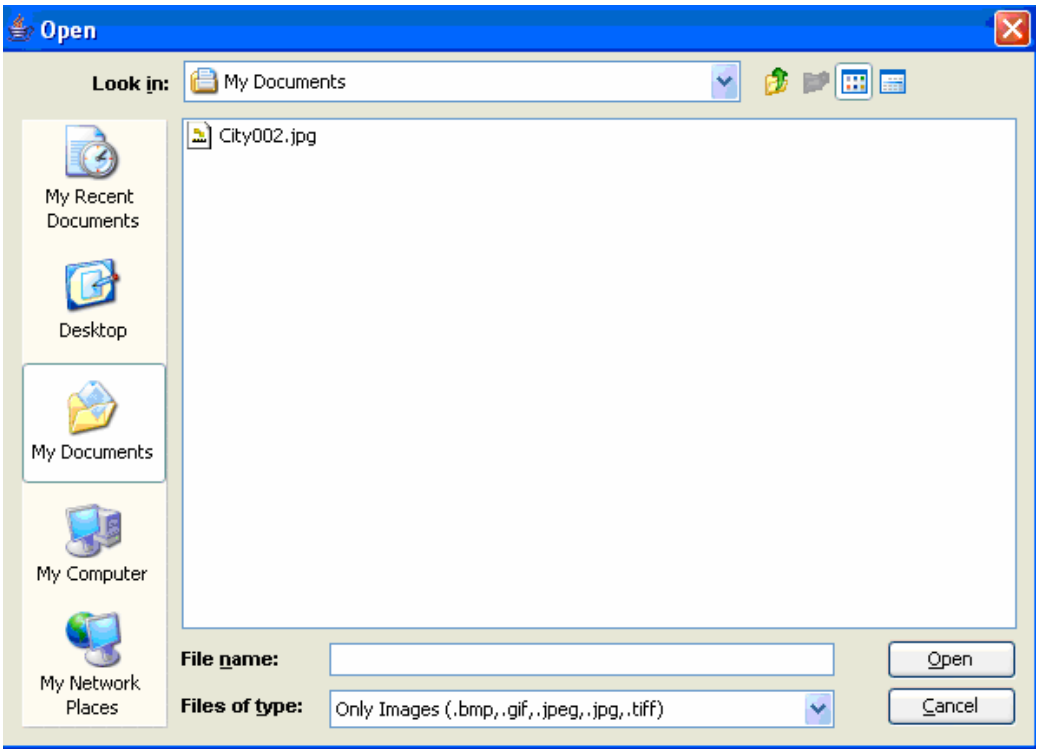


Figure 32 – Choose image dialog box

The user can only select files with extension .gif, .jpg, .jpeg, .bmp and .tiff. Once the user has selected the image, he has to click on the "Apply changes" button for the images to be appended to the particular question. Clicking on the "Apply Changes" button displays the pathname of the image file just below the question as shown in Figure 33.

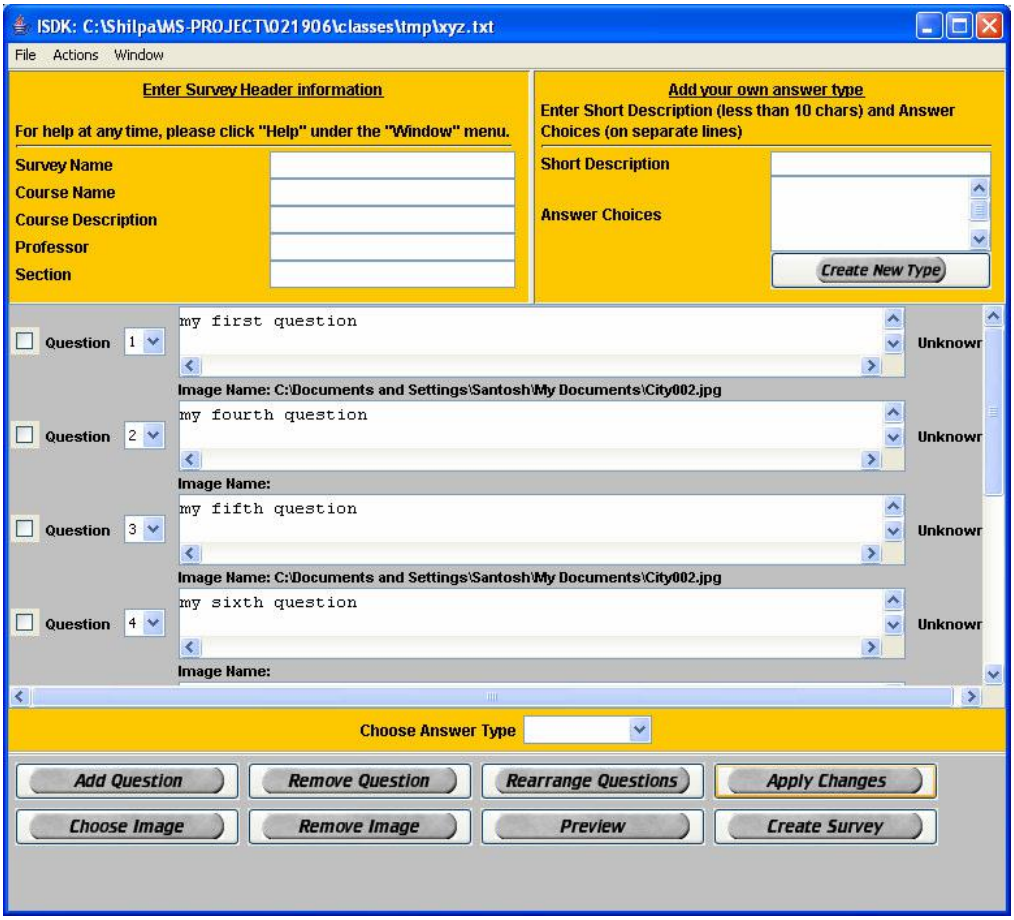


Figure 33 – Demonstration of attaching images

The “Remove Image” feature allows the user to delete an image that has been appended to a question. The user can either delete a single question or multiple questions at once. The user has to select the questions for which the images have to be deleted and then click on "Remove Image" button. For example, if the user wants to delete the image from the third question, he has to check the box next to the third question and clicking on “Remove Image” button deletes the image as shown in Figure 34.

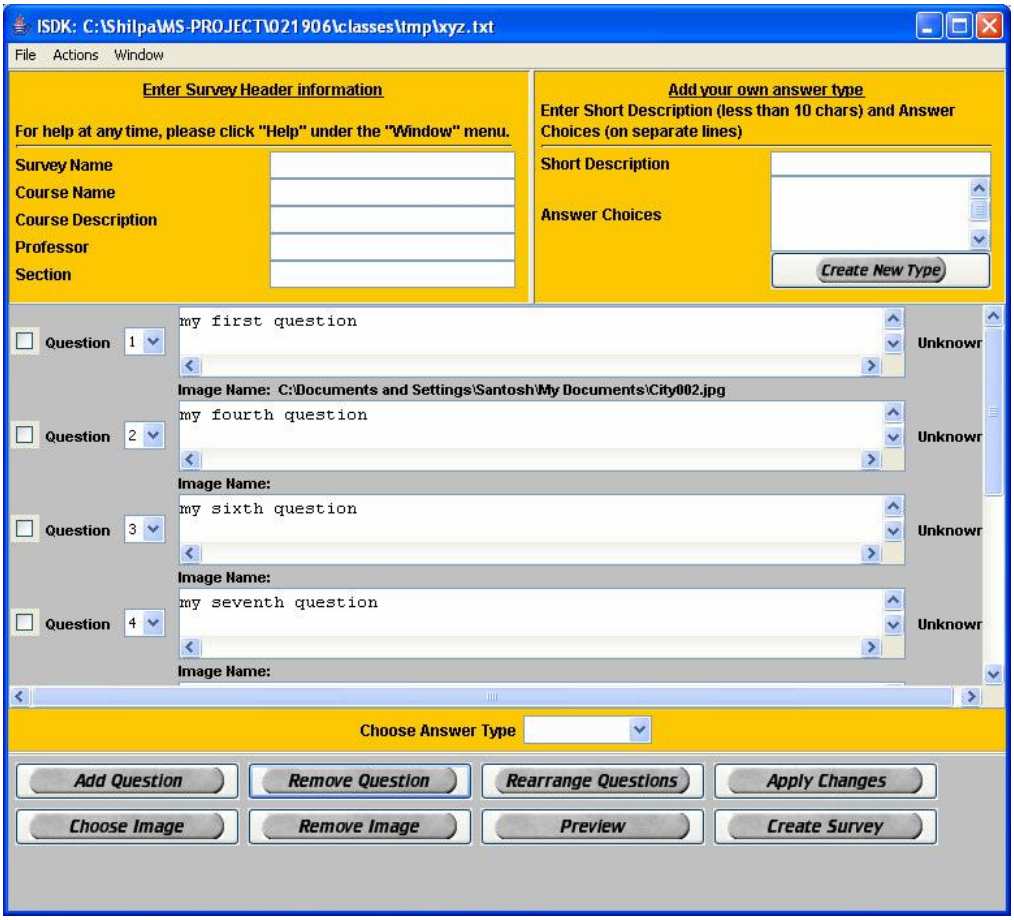


Figure 34 – Demonstration of deleting images

If the user wants to quit the application half-way through and resume it at a later time, he can click on the “Save Survey” option under the File menu as shown in Figure 35.

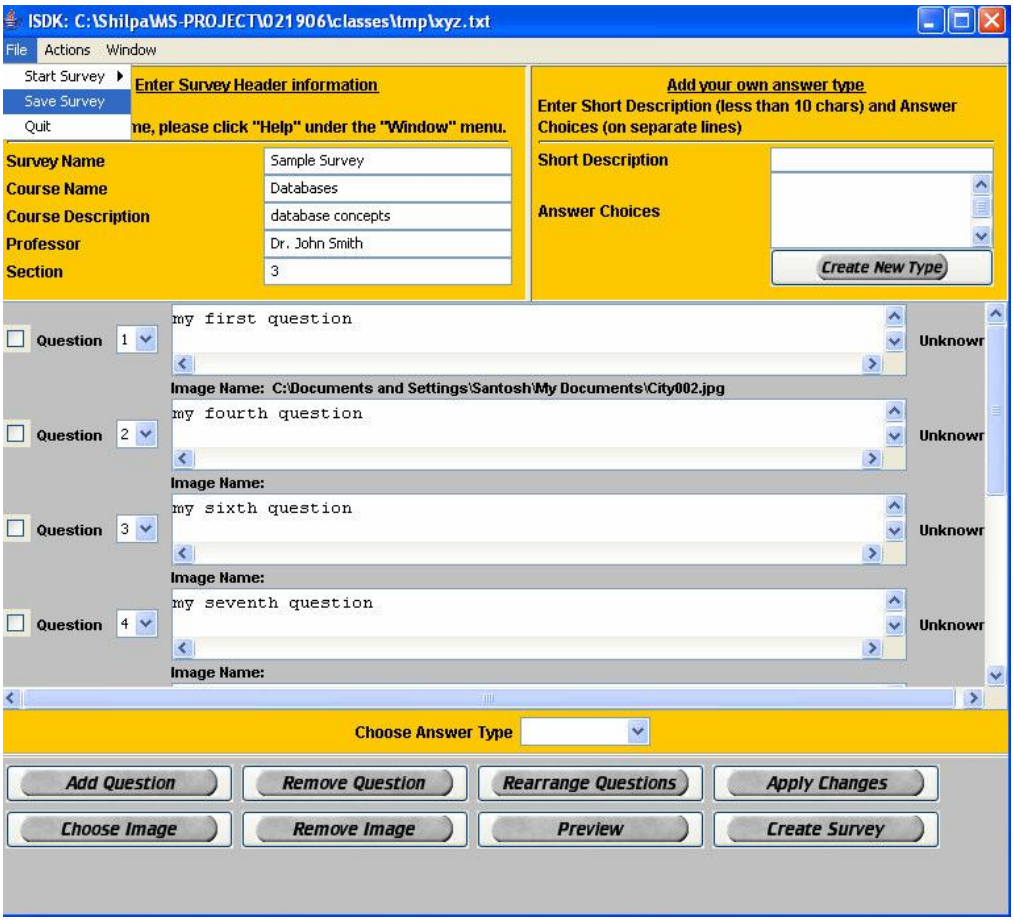


Figure 35 – Demonstration of saving a survey

Doing this will save all the questions along with the answer choices that have been selected so far so that he does not have to start all over again. The next time he wants to resume working on the survey, he has to select the survey using the "Upload Past Survey" option under File menu. Clicking on this option opens a "Choose Past Survey File" dialog box as shown in Figure 36.

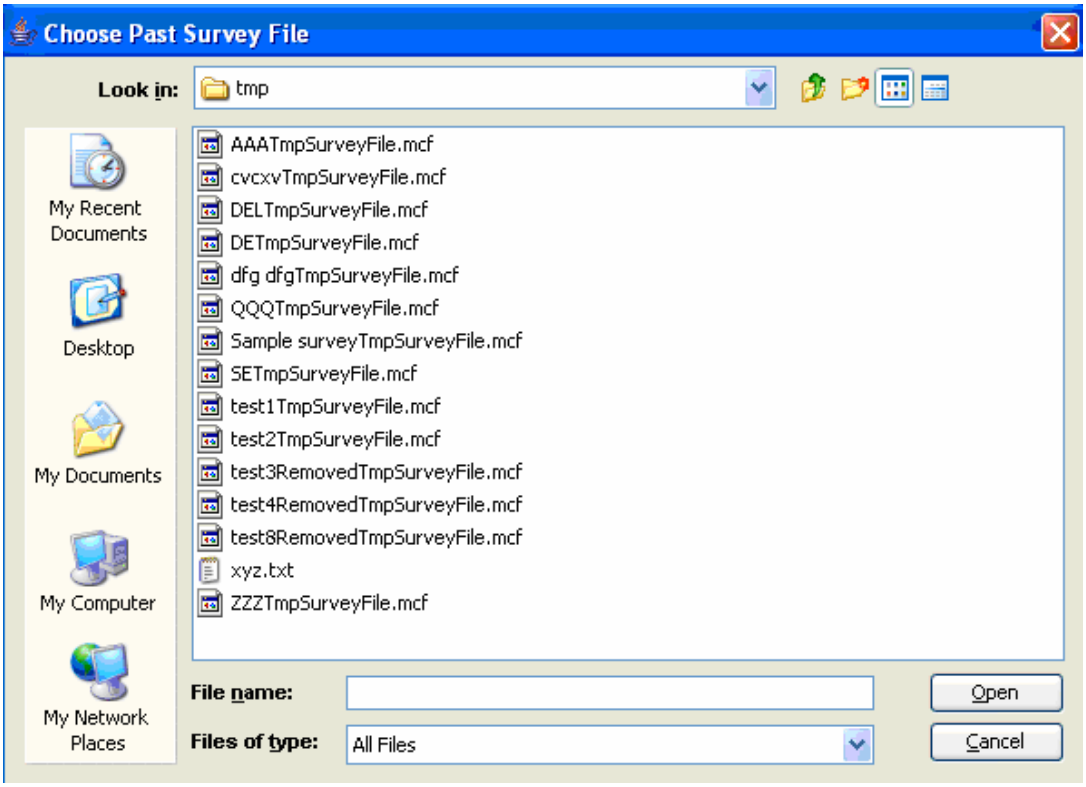


Figure 36– Choose past survey file dialog box

Once the user has selected the file, the survey is presented in a format that will allow him to resume from the point where he left off. The header information along with all the questions and the answers selected so far are retained so that the user does not have to re-enter them again.

The “Preview” feature allows the user to view the survey page even before creating the actual survey. This feature helps the user in making modifications before creating the survey. In order to preview the survey, it is not necessary for the user to have completed working with all the questions. If the user has selected answer choices only for some of the questions, in the preview the answer choices for the remaining questions will show up as unknown. A sample preview is shown in Figure 37.

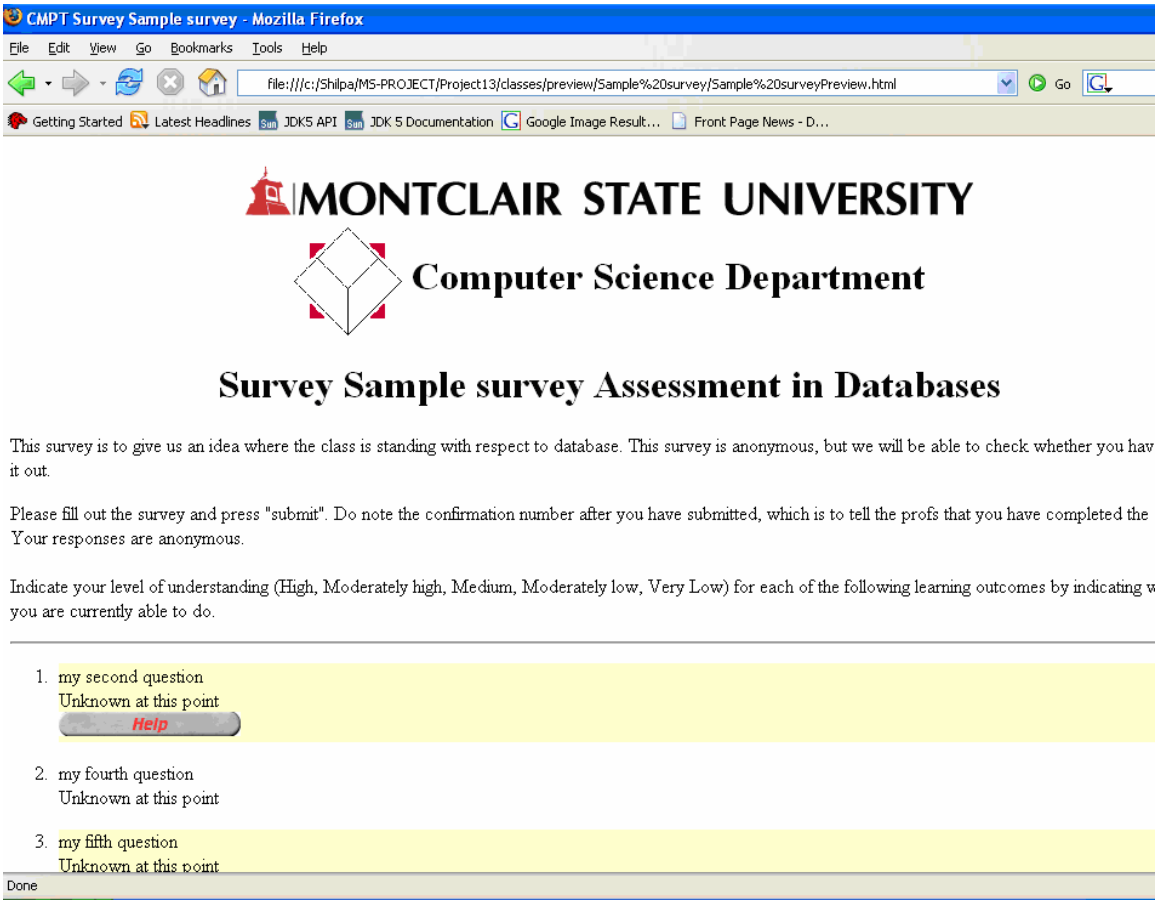


Figure 37 – Sample preview of a survey

The “Create Survey” feature is used to create the actual survey files namely the formatted text file, the HTML file and the JSP file. Once the survey files are created, a confirmation message appears on the screen specifying the location of the survey files as shown in Figure 38.

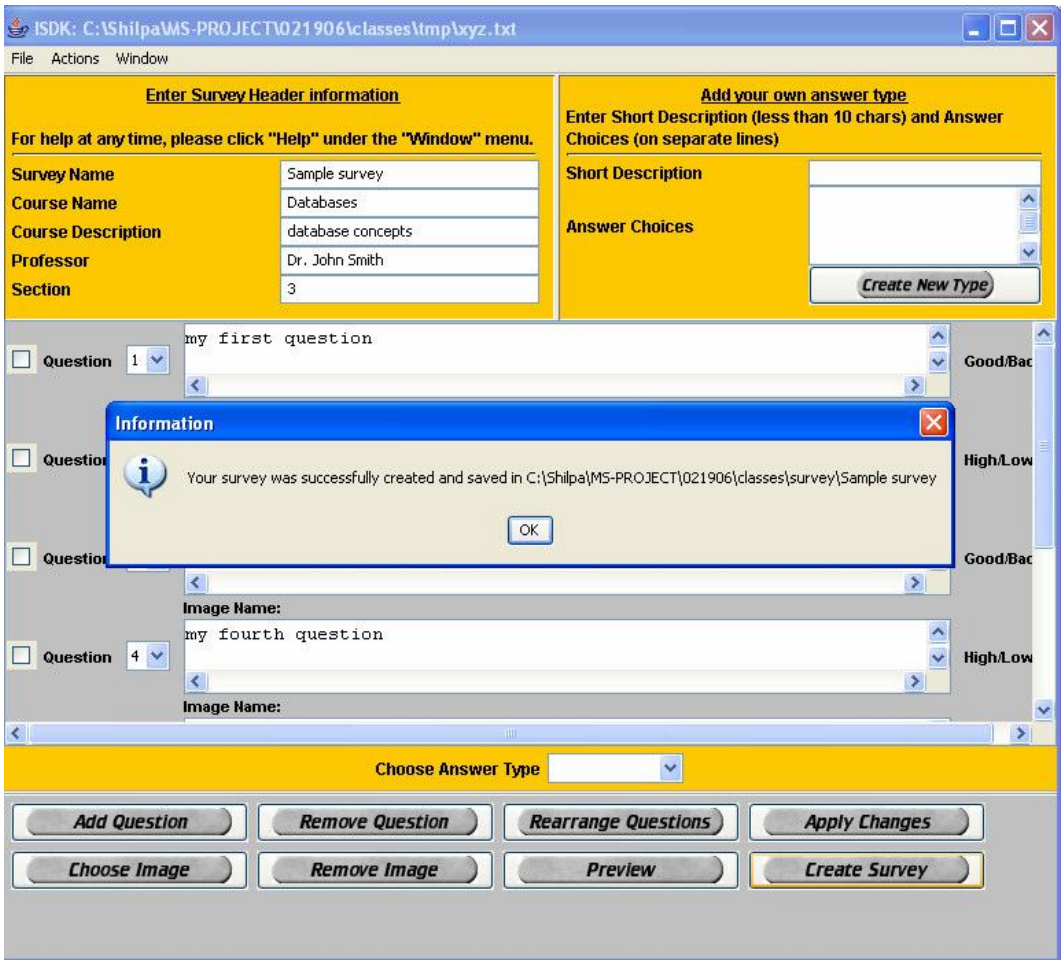


Figure 38 – Survey creation confirmation message

There is a "Help" feature under Windows menu that provides information on the usage of all the features. Clicking on the Help menu option displays a screen as shown in Figure 39.

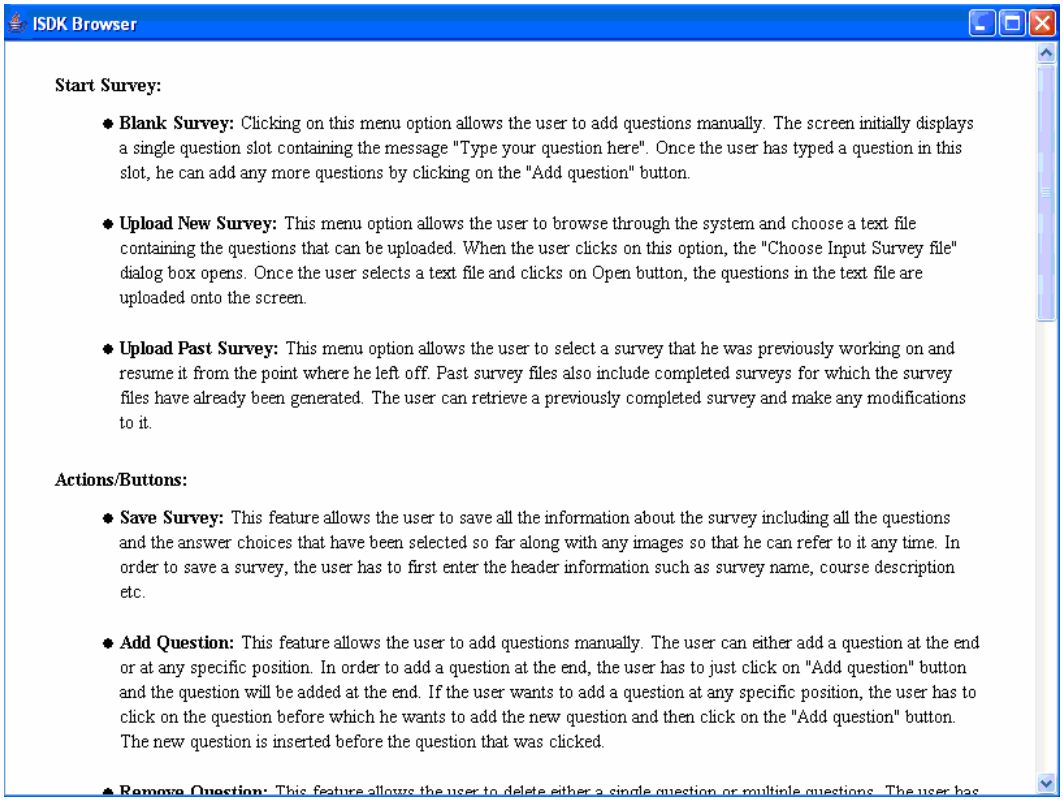


Figure 39 – Help window

7. Conclusion

Web-based surveys have had a profound influence on the survey process in a number of ways. The survey taking process has become more democratized as a result of Web surveys. Web surveys are a visual stimulus, and respondents have control over how and even whether they read and comprehend each question. Unlike other types of surveys, Web page design skills and computer programming expertise play a significant role in the design of Web-based surveys.

In this project, we have designed and implemented a survey development tool that is easy to use and flexible with regard to question and layout design. The primary advantage of the application is the automation of the survey creation mechanism developed within the Computer Science department to support data collection. The current state of the art requires the person maintaining and updating the system to perform significant work in file creation and editing. Through this application, we have succeeded in automating the survey creation process such that the user himself can create the survey files without having to indulge in file creation and editing.

References

[1] Gunn H., "Web-based Surveys: Changing the Survey Process", *First Monday*, Vol 7, 12, December 2002, http://www.firstmonday.org/issues/issue7_12/gunn/, accessed March 15, 2006

[2] Carini R.M., Hayek J.C., Kuh G.D., Kennedy J.M., and Quimet J. A., "College Students Responses to Web and Paper Surveys: Does Mode Matter?", *Research in Higher Education,* Vol. 44, 1, February 2003, http://nsse.iub.edu/pdf/mode.pdf, accessed March 15, 2006

[3] Daly, R. F., Cross, J., and Thomson, G. E. "Web versus paper surveys: Results of a large-scale direct comparison of the two methods. Paper presented at the Forum of the Association for Institutional Research, Long Beach, CA, 2001 (cited in [2])

[4] Bredlau C., "Web Based Survey System", *unpublished documentation*, 2005

[5] ***, *Montclair State University - Survey System***, https://surveys.montclair.edu/survey/login.jsp?r=1, accessed November 1, 2005

[6] ***, *Virginia Tech University Survey System (VTSurvey)*, 2005**,** http://vtsurvey.sourceforge.net/index.php , accessed November 1, 2005

[7] ***, Blackboard Inc. Company Description, 2006, http://www.blackboard.com/company/, accessed March 13, 2006

[8] ***, *Blackboard Survey Manager – Blackboard v. 6.1*, 2004 http://www.blackboard.com/docs/r6/6_1/instructor/bbls_r6_1_instructor/survey_manager.htm, accessed March 13, 2006

[9] Sun Microsystems (2005). http://java.sun.com/docs/books/tutorial/uiswing/index.html

[10] Deitel, H. M., and Deitel, P. J., *Java How to Program*, Fifth Edition, New Delhi: Pearson Education, 2004

[11] Lewis, John and Loftus, William, *Java Software Solutions*, Third Edition, United States of America: Addison Wesley, 2003

[12] The Johns Hopkins University (1999) *Swing: A Quick Tutorial for AWT Programmers* http://www.apl.jhu.edu/~hall/java/Swing-Tutorial/Swing-Tutorial-Intro.html

[13] Fujaba Tool Suite 4.0.1 (2003). http://wwwcs.uni-paderborn.de/cs/fujaba/downloads/index.html

[14] Cooltext – An online Logo Generator (2006). http://cooltext.com/LogoEdit.aspx?Style=Blended