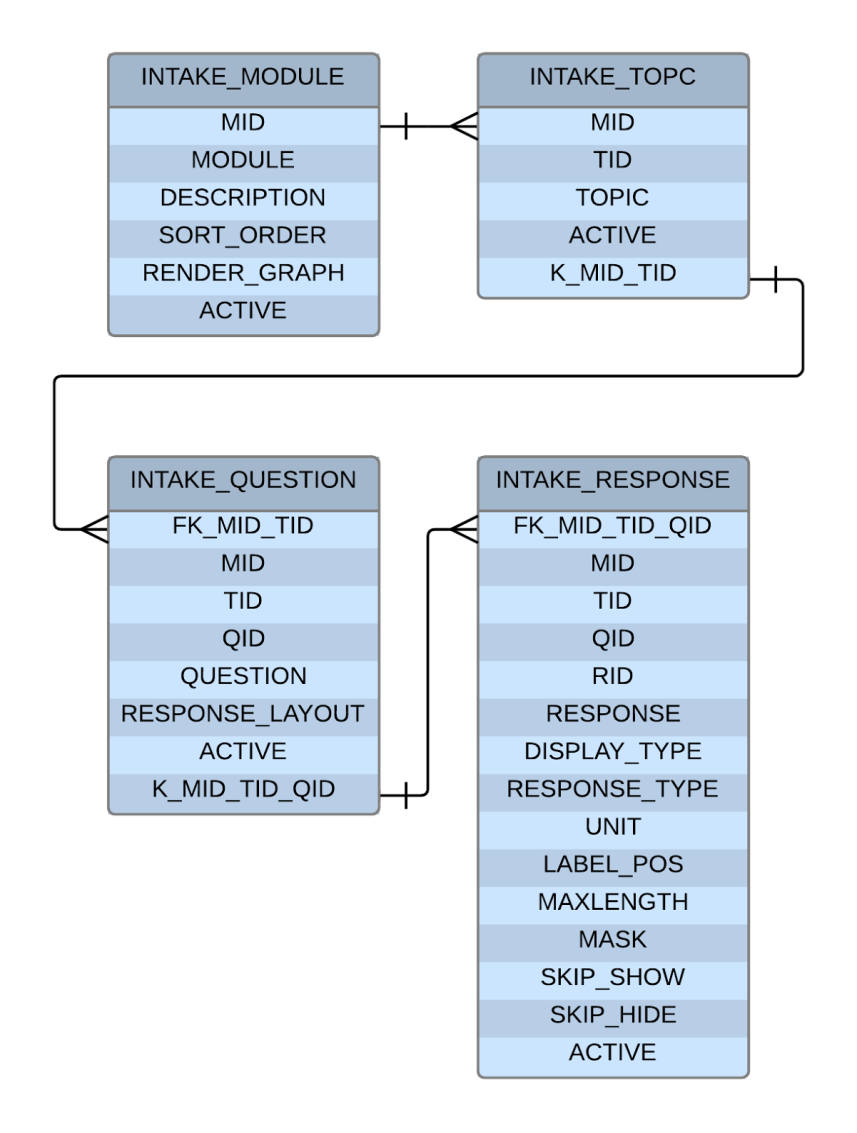
# Design Document for User Story 5767

### As a user I want instrument questions and responses stored in the TBI central database so that they are easy to maintain and query.

## Dynamically Generated Instruments (Intakes)

### Data Model (Simplified)

Tables and Relationships



### Packages and Stored Procedures

#### PCK\_QUESTIONNAIRES

Stored Procedures:

* GetModuleRS
  + IN Parameters
    - pi\_nMID (number)
      * The module id
  + OUT Parameters
    - rs (cursor)
      * The module recordset
* GetTopicRS
  + IN Parameters
    - pi\_nMID (number)
      * The module id
  + OUT Parameters
    - rs (cursor)
      * The topic recordset for the module
* GetQuestionRS
  + IN Parameters
    - pi\_nMID (number)
      * The module id
    - pi\_nTID (number)
      * The topic id
  + OUT Parameters
    - rs (cursor)
      * The question recordset for the module, topic
* GetResponseRS
  + IN Parameters
    - pi\_nMID (number)
      * The module id
    - pi\_nTID (number)
      * The topic id
    - pi\_nQID (number)
      * The question id
  + OUT Parameters
    - rs (cursor)
      * The question recordset for the module, topic, question
* GetSkipPatternRS
  + IN Parameters
    - pi\_nMID (number)
      * The module id
  + OUT Parameters
    - rs (cursor)
      * The question recordset for the response’s skip rules.
* GetMaskRS
  + IN Parameters
    - pi\_nMID (number)
      * The module id
  + OUT Parameters
    - rs (cursor)
      * The question recordset for the response’s mask rules.

### C# Class(es)

#### CQuestionnaire.cs

Properties

* BaseMaster m\_BaseMstr (public)

Methods

* public DataSet GetModuleDS(long lMID)
  + Returns a module DataSet from PCK\_QUESTIONNAIRE.GetModuleRS for the passed on module id.
* public DataSet GetModuleTopicDS(long lMID)
  + Returns a topic DataSet from PCK\_QUESTIONNAIRE.GetTopicRS for the passed on module id.
* public DataSet GetQuestionDS(long lMID, long lTID)
  + Returns a question DataSet from PCK\_QUESTIONNAIRE.GetQuestionRS for the passed on module, topic.
* public DataSet GetResponseDS(long lMID, long lTID, long lQID)
  + Returns a response DataSet from PCK\_QUESTIONNAIRE.GetResponseRS for the passed on module, topic, question.
* public DataSet GetSkipPatternDS(long lMID)
  + Returns a topic DataSet from PCK\_QUESTIONNAIRE.GetSkipPatternRS for the passed on module.
* public DataSet GetMaskDS(long lMID)
  + Returns a topic DataSet from PCK\_QUESTIONNAIRE.GetMaskRS for the passed on module.
* public string GetQuestionnaire(long lMID)
  + Returns a XML string with all the module’s data:
    - module -> topic -> question -> response
* public string GetSkipPattern(long lMID)
  + Calls this.GetSkipPatternDS(long lMID). From the returned dataset, builds and returns a JSON string with the skip pattern rules for the module.
* public string GetMask(long lMID)
  + Calls this.GetMaskDS(long lMID). From the returned dataset, builds and returns a JSON string with the mask rules for the module.

### Generated XML and XSL Transformation

#### XML Schema

This is a structural representation (simplified) of the generated XML for the module.

(See the XSD in Appendix A.)

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <modules>  <module>  <mid />  <module\_title />  <topics>  <topic>  <tid />  <topic\_title />  <questions>  <question>  <qid />  <question\_text />  <responses>  <response>  <rid />  <response\_text />  <display\_type />  </response>  </responses>  </question>  </questions>  </topic>  </topics>  </module>  </modules> |

#### XSL Transformation

(See Appendix B)

#### XML Sample (for module 1000)

(See Appendix C)

### ASP.NET Page and Code-behind

#### ASP.NET Page

1. Contents of the XML will be dumped into an asp:xml control:
   1. <asp:Xml ID="xmlQuestionnaire" runat="server"></asp:Xml>
2. Skip Pattern and Mask JSON string will be used as literals inside the <script /> tag on the page.
   1. The string are being parsed to object by using this JS function: JSON.parse(string);

##### Code-behind

Code-behind for this page has only to main functions:

1. Render the Instrument
   1. Get the XML for the module
   2. Transform the XML using a Stylesheet
   3. Get the Skip Pattern and Mask JSON strings
2. Process the responses and write to the DB (Nothing has been implemented yet)

### Client-side scripting

The page depends on a JS object that is responsible of traverse the DOM for:

1. Bind events to the responses:
   1. Triggers for the skip rules
   2. Keyup checks for the textbox masks
   3. Type99 (clearAbove) associated events
2. Toggle visibility for the Skip Rules
3. Check for required answers before submitting the responses
   1. display particular alerts in the question with validation issues

Current question.js code is included in Appendix D.

## Appendices

### Appendix A - XSD (XML Schema)

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">  <xs:element name="MODULES">  <xs:complexType>  <xs:sequence>  <xs:element name="MODULE">  <xs:complexType>  <xs:sequence>  <xs:element name="MID" type="xs:int" />  <xs:element name="MODULE\_TITLE" type="xs:string" />  <xs:element name="ACTIVE" type="xs:int" />  <xs:element name="DESCRIPTION" type="xs:string" />  <xs:element name="SORT\_ORDER" type="xs:int" />  <xs:element name="RENDER\_GRAPH" type="xs:int" />  <xs:element name="TOPICS">  <xs:complexType>  <xs:sequence>  <xs:element name="TOPIC" maxOccurs="unbounded">  <xs:complexType>  <xs:sequence>  <xs:element name="MID" type="xs:int" />  <xs:element name="TID" type="xs:int" />  <xs:element name="TOPIC" type="xs:string" />  <xs:element name="ACTIVE" type="xs:int" />  <xs:element name="DEL\_ALTERNATE\_LANGUAGE" type="xs:string" />  <xs:element name="QUESTIONS">  <xs:complexType>  <xs:sequence>  <xs:element name="QUESTION" maxOccurs="unbounded">  <xs:complexType>  <xs:sequence>  <xs:element name="MID" type="xs:int" />  <xs:element name="TID" type="xs:int" />  <xs:element name="QID" type="xs:int" />  <xs:element name="QUESTION" type="xs:string" />  <xs:element name="ACTIVE" type="xs:int" />  <xs:element name="RESPONSES\_POS" type="xs:int" />  <xs:element name="RESPONSES">  <xs:complexType>  <xs:sequence>  <xs:element name="RESPONSE" maxOccurs="unbounded">  <xs:complexType>  <xs:sequence>  <xs:element name="RID" type="xs:int" />  <xs:element name="MID" type="xs:int" />  <xs:element name="TID" type="xs:int" />  <xs:element name="QID" type="xs:int" />  <xs:element name="RESPONSE\_TEXT" type="xs:string" />  <xs:element name="DISPLAY\_TYPE" type="xs:int" />  <xs:element name="RESPONSE\_TYPE" type="xs:int" />  <xs:element name="ACTIVE" type="xs:int" />  <xs:element name="RESPONSE\_GROUP" type="xs:int" />  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:schema> |

### Appendix B - Instrument XSL (Stylesheet)

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:msxsl="urn:schemas-microsoft-com:xslt" version="1.0" exclude-result-prefixes="msxsl">  <xsl:output method="xml" indent="yes" />  <xsl:template match="/MODULES">  <xsl:for-each select="MODULE">  <!-- Module Title -->  <div class="container">  <div class="page-header">  <h3>  <xsl:value-of select="MODULE\_TITLE" />  </h3>  </div>  <!-- Iterate Topics -->  <xsl:for-each select="TOPICS/TOPIC">  <div class="panel panel-primary">  <div class="panel-heading">  <h3 class="panel-title">  <xsl:value-of select="TOPIC" />  </h3>  </div>  <div class="panel-body">  <!-- Iterate Questions -->  <xsl:for-each select="QUESTIONS/QUESTION">  <div id="TID{TID}\_QID{QID}" class="question-container">  <h4>  <small>  TID  <xsl:value-of select="TID" />  \_QID  <xsl:value-of select="QID" />  </small>    <xsl:value-of select="QUESTION" />  </h4>  <!-- Iterate Responses -->  <xsl:choose>  <xsl:when test="RESPONSES\_POS = 1">  <ul style="list-style: none;">  <xsl:apply-templates select="RESPONSES" />  </ul>  </xsl:when>  <xsl:otherwise>  <ul class="responses-horizontal" style="list-style: none;">  <xsl:apply-templates select="RESPONSES" />  </ul>  </xsl:otherwise>  </xsl:choose>  </div>  </xsl:for-each>  </div>  </div>  </xsl:for-each>  </div>  </xsl:for-each>  </xsl:template>  <xsl:template match="RESPONSES">  <xsl:for-each select="RESPONSE">  <li>  <xsl:choose>  <!-- Render Radio buttons -->  <xsl:when test="DISPLAY\_TYPE = 1">  <label>  <input type="radio" id="rid\_{RID}" name="grpRadio\_{RESPONSE\_GROUP}" value="{RESPONSE}|{RID}|{SCORE\_VALUE}" />    <xsl:value-of select="RESPONSE\_TEXT" />  </label>  </xsl:when>  <!-- Render Check boxes -->  <xsl:when test="DISPLAY\_TYPE = 2">  <label>  <input type="checkbox" id="rid\_{RID}" name="grpCheck\_{RESPONSE\_GROUP}" value="{RESPONSE}|{RID}|{SCORE\_VALUE}" />    <xsl:value-of select="RESPONSE\_TEXT" />  </label>  </xsl:when>  <!-- Render Text box -->  <xsl:when test="DISPLAY\_TYPE = 3">  <label>  <xsl:value-of select="RESPONSE\_TEXT" />    <input type="text" id="rid\_txt\_{RID}" name="grpCtrlText\_{RESPONSE\_GROUP}" />    <xsl:value-of select="UNIT" />  </label>  <input type="hidden" name="grpHidden\_{RESPONSE\_GROUP}" id="rid\_{RID}" value="|{RID}|{SCORE\_VALUE}" />  </xsl:when>  <!-- Render 'None of the above' Check boxes -->  <xsl:when test="DISPLAY\_TYPE = 99">  <label>  <input type="checkbox" id="rid\_{RID}" name="grpCheck\_{RESPONSE\_GROUP}" clearabove="clearabove" value="{RESPONSE}|{RID}|{SCORE\_VALUE}" />    <xsl:value-of select="RESPONSE\_TEXT" />  </label>  </xsl:when>  <xsl:otherwise>  <xsl:value-of select="RESPONSE\_TEXT" />  </xsl:otherwise>  </xsl:choose>  </li>  </xsl:for-each>  </xsl:template>  </xsl:stylesheet> |

### Appendix C - Sample XML

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:msxsl="urn:schemas-microsoft-com:xslt" version="1.0" exclude-result-prefixes="msxsl">  <xsl:output method="xml" indent="yes" />  <xsl:template match="/MODULES">  <xsl:for-each select="MODULE">  <!-- Module Title -->  <div class="container">  <div class="page-header">  <h3>  <xsl:value-of select="MODULE\_TITLE" />  </h3>  </div>  <!-- Iterate Topics -->  <xsl:for-each select="TOPICS/TOPIC">  <div class="panel panel-primary">  <div class="panel-heading">  <h3 class="panel-title">  <xsl:value-of select="TOPIC" />  </h3>  </div>  <div class="panel-body">  <!-- Iterate Questions -->  <xsl:for-each select="QUESTIONS/QUESTION">  <div id="TID{TID}\_QID{QID}" class="question-container">  <h4>  <small>  TID  <xsl:value-of select="TID" />  \_QID  <xsl:value-of select="QID" />  </small>    <xsl:value-of select="QUESTION" />  </h4>  <!-- Iterate Responses -->  <xsl:choose>  <xsl:when test="RESPONSES\_POS = 1">  <ul style="list-style: none;">  <xsl:apply-templates select="RESPONSES" />  </ul>  </xsl:when>  <xsl:otherwise>  <ul class="responses-horizontal" style="list-style: none;">  <xsl:apply-templates select="RESPONSES" />  </ul>  </xsl:otherwise>  </xsl:choose>  </div>  </xsl:for-each>  </div>  </div>  </xsl:for-each>  </div>  </xsl:for-each>  </xsl:template>  <xsl:template match="RESPONSES">  <xsl:for-each select="RESPONSE">  <li>  <xsl:choose>  <!-- Render Radio buttons -->  <xsl:when test="DISPLAY\_TYPE = 1">  <label>  <input type="radio" id="rid\_{RID}" name="grpRadio\_{RESPONSE\_GROUP}" value="{RESPONSE}|{RID}|{SCORE\_VALUE}" />    <xsl:value-of select="RESPONSE\_TEXT" />  </label>  </xsl:when>  <!-- Render Check boxes -->  <xsl:when test="DISPLAY\_TYPE = 2">  <label>  <input type="checkbox" id="rid\_{RID}" name="grpCheck\_{RESPONSE\_GROUP}" value="{RESPONSE}|{RID}|{SCORE\_VALUE}" />    <xsl:value-of select="RESPONSE\_TEXT" />  </label>  </xsl:when>  <!-- Render Text box -->  <xsl:when test="DISPLAY\_TYPE = 3">  <label>  <xsl:value-of select="RESPONSE\_TEXT" />    <input type="text" id="rid\_txt\_{RID}" name="grpCtrlText\_{RESPONSE\_GROUP}" />    <xsl:value-of select="UNIT" />  </label>  <input type="hidden" name="grpHidden\_{RESPONSE\_GROUP}" id="rid\_{RID}" value="|{RID}|{SCORE\_VALUE}" />  </xsl:when>  <!-- Render 'None of the above' Check boxes -->  <xsl:when test="DISPLAY\_TYPE = 99">  <label>  <input type="checkbox" id="rid\_{RID}" name="grpCheck\_{RESPONSE\_GROUP}" clearabove="clearabove" value="{RESPONSE}|{RID}|{SCORE\_VALUE}" />    <xsl:value-of select="RESPONSE\_TEXT" />  </label>  </xsl:when>  <xsl:otherwise>  <xsl:value-of select="RESPONSE\_TEXT" />  </xsl:otherwise>  </xsl:choose>  </li>  </xsl:for-each>  </xsl:template>  </xsl:stylesheet> |

### Appendix D - JavaScript Object “question.js”

|  |
| --- |
| var question = {  // GENERAL SETTINGS  opts: {  debug\_mode: true,  containers: {  questionnaire: {  id: 'divQuestions'  }  }  },  // fn RESPONSES  clearQuestionResponses: function (divQestion) {  var \_me = this;  //clear previous validation error messages  //\_me.clearPrevErrors(divQestion);  // clear radio and checkboxes  $('input[type="radio"], input[type="checkbox"]', $(divQestion)).each(function () {  this.checked = false;  });  // clear textboxes  $('input[type="text"], input[type="hidden"]', $(divQestion)).each(function () {  this.value = null;  });  },  validateQuestionnaire: function () {  var \_me = this,  \_qDiv = $('div.question-container').not('[skipped]'),  \_qCount = \_qDiv.length,  \_rCount = 0;  $.each(\_qDiv, function (i, d) {  var c1 = $('input[type="radio"]:checked, input[type="checkbox"]:checked', $(d)).length,  c2 = $('select', $(d)).filter(function (i, ele) {  return ele.selectedIndex > 0;  }).length,  c3 = $('input[type="text"]', $(d)).filter(function (i, ele) {  return ele.value.length > 0;  }).length;  //clear previous validation error messages  \_me.clearPrevErrors(d);  if ((c1 + c2 + c3) > 0) {  \_rCount += 1;  } else {  //show error message  \_me.showValidationErrMessage(d);  }  });  if (\_qCount > 0) {  if (\_rCount >= \_qCount) {  return true;  } else {  alert('Please review highlighted question, make necessary corrections and submit your responses again.');  return false;  }  }  return false;  },  showValidationErrMessage: function (div) {  var \_me = this,  \_msg = '<h5 class="validation-err" style="color: red;"><span class="glyphicon glyphicon-warning-sign"></span>&nbsp;A response is required for this question.</h5>';  $(div).addClass('bg-warning');  $('h4', $(div)).prepend(\_msg);  },  clearPrevErrors: function (div) {  var \_me = this;  //clear validation error warnings  $('h5.validation-err', $(div)).remove();  $(div).removeClass('bg-warning');  },  clearAbove: function (ele) {  var \_me = this,  \_dQ = \_me.getQuestionDivByRID(ele.id);  //check if the element has the 'clearabove' attribute  if (ele.getAttribute('clearabove') != null) {  if (ele.checked) {  //clear question responses  if (\_dQ) {  \_me.clearQuestionResponses(\_dQ);  ele.checked = true;  }  }  } else {  // ----- check for the related controls status -----  if (ele.tagName.toLowerCase() == 'input') {  var \_type = $(ele).attr('type').toLowerCase();  switch (\_type) {  case 'radio':  if (ele.checked) {  if (\_dQ) {  $('[clearabove]', $(\_dQ)).attr('checked', false);  }  }  break;  case 'checkbox':  if (ele.checked) {  if (\_dQ) {  $('[clearabove]', $(\_dQ)).attr('checked', false);  }  }  break;  case 'text':  if ($(ele).val().length > 0) {  if (\_dQ) {  $('[clearabove]', $(\_dQ)).attr('checked', false);  }  }  break;  }  } else if (ele.tagName == 'select') {  //do something  } else if (ele.tagName == 'textarea') {  //do something  }  }  },  // fn SKIP PATTERNS  checkSkipPatterns: function () {  var \_me = this;  if (typeof (\_me.opts.skipPatterns) === "object") {  //iterate the skip rules  $.each(\_me.opts.skipPatterns, function (a, b) {  //get the skip trigger control  if (!isNaN(b.rid)) {  \_me.checkSimpleRIDSkip(b);  } else {  return;  }  });  }  },  checkSimpleRIDSkip: function (b) {  var \_me = this;  var r = $('input[type="checkbox"], input[type="radio"]').filter(function (i, ele) {  return $(this).attr('id') == "rid\_" + b.rid;  })[0];  if (typeof (r) !== "undefined") {  if (r.checked) {  //show questions  if (typeof (b.checked.show) !== "undefined") {  b.checked.show.forEach(function (qID) {  var \_divQ = $('div[id$="' + qID + '"]');  //remove 'skipped' attribute  $(\_divQ).removeAttr('skipped');  //reset question's responses  \_me.clearQuestionResponses(\_divQ);  //show question  \_divQ.show();  });  }  //hide questions  if (typeof (b.checked.hide) !== "undefined") {  b.checked.hide.forEach(function (qID) {  var \_divQ = $('div[id$="' + qID + '"]');  //set 'skipped' attribute  $(\_divQ).attr('skipped', true);  //reset question's responses  \_me.clearQuestionResponses(\_divQ);  //hide question  \_divQ.hide();  });  }  } else {  //hide questions  if (typeof (b.checked.show) !== "undefined") {  b.checked.show.forEach(function (qID) {  var \_divQ = $('div[id$="' + qID + '"]');  //set 'skipped' attribute  $(\_divQ).attr('skipped', true);  //reset question's responses  \_me.clearQuestionResponses(\_divQ);  //hide question  \_divQ.hide();  });  }  //show questions  if (typeof (b.checked.hide) !== "undefined") {  b.checked.hide.forEach(function (qID) {  var \_divQ = $('div[id$="' + qID + '"]');  //remove 'skipped' attribute  $(\_divQ).removeAttr('skipped');  //reset question's responses  \_me.clearQuestionResponses(\_divQ);  //show question  \_divQ.show();  });  }  }  }  },  //MASKS  mask: {  numbersOnly: function (obj) {  var p = question,  me = this,  id = obj.id.replace(/\D/gi, ''),  m = null,  dec = false,  reNum = /[^0-9.]/gi;  if (typeof (p.opts.txtMasks) != "undefined") {  m = p.opts.txtMasks;  $.each(m, function (i, o) {  if (o.rid == id) {  if (typeof (o.allowDecimal) != "undefined") {  dec = o.allowDecimal;  if (o.allowDecimal) {  reNum = /[^0-9.]/gi;  } else {  reNum = /\D/gi;  }  }  }  });  }  obj.value = obj.value.replace(reNum, '');  if (dec) {  var val = obj.value,  val2 = val.replace(/[^.]/gi, '');  if (val2.length > 1) {  obj.value = val.substr(0, val.length - 1);  }  }  },  //apply HHmm24h  HHmm24h: function (obj) {  var me = this,  test = [  '(\\d)',  '([01]\\d|2[0-3])|([1-9]:)',  '([01]\\d:|2[0-3]:)|([1-9]:[0-5])',  '(([01]\\d|2[0-3]):([0-5]))|([1-9]:[0-5]\\d)',  '(([01]\\d|2[0-3]):([0-5]\\d))'  ];  if (obj.value.length > test.length) {  obj.value = obj.value.substr(0, obj.value.length - 1);  return false;  } else if (obj.value.length == 0) {  return true;  } else {  var reTest = new RegExp(test[obj.value.length - 1]);  if (reTest.test(obj.value)) {  return true;  } else {  obj.value = obj.value.substr(0, obj.value.length - 1);  return false;  }  }  }  },  // BINDINGS ---------------------------  bindSkipPatterns: function () {  var \_me = this;  if (typeof (\_me.opts.skipPatterns) === "object") {  //iterate the skip rules  $.each(\_me.opts.skipPatterns, function (a, b) {  if (!isNaN(b.rid)) {  //get the question div of the response with skip  var \_div = \_me.getQuestionDivByRID('rid\_' + b.rid);  //bind the skip check to the responses  if (\_div) {  $('input[type="radio"], input[type="checkbox"]', \_div).each(function () {  var b = $(this).unbind('click').bind('click', function (e) {  \_me.checkSkipPatterns(e);  });  if (\_me.opts.debug\_mode) {  $(b).parent().css({  color: 'red'  });  }  });  }  }  });  }  return;  },  bindClearAbove: function () {  var \_me = this;  $('[clearabove]').each(function () {  var rid = this.id,  \_dQuestion = \_me.getQuestionDivByRID(rid);  if (typeof (\_dQuestion) !== "undefined") {  //bind radio and checkboxes  $('input[type="radio"], input[type="checkbox"]', $(\_dQuestion)).bind('click', function (e) {  \_me.clearAbove(this);  });  //bind dropdown  $('select', $(\_dQuestion)).bind('change', function (e) {  \_me.clearAbove(this);  });  //bind textbox and textarea  $('input[type="text"]', $(\_dQuestion)).bind('keyup', function (e) {  \_me.clearAbove(this);  });  }  });  },  //apply masks to textboxes  applyMask: function () {  var \_me = this;  if (typeof (\_me.opts.txtMasks) != "undefined") {  $.each(\_me.opts.txtMasks, function (i, o) {  var \_txt = $('input[type="text"][id="rid\_txt\_' + o.rid + '"]')[0];  if (typeof (o.maxlength) != "undefined") {  $(\_txt).attr('maxlength', o.maxlength);  }  if (typeof (o.mask) != "undefined") {  if (o.mask.toLowerCase() == "numbersonly") {  var \_onkeyup = $(\_txt).attr('onkeyup');  $(\_txt).attr('onkeyup', 'question.mask.numbersOnly(this); ' + \_onkeyup);  }  }  if (typeof (o.mask) != "undefined") {  if (o.mask.toLowerCase() == "hhmm24h") {  var \_onkeyup = $(\_txt).attr('onkeyup');  $(\_txt).attr('onkeyup', 'question.mask.HHmm24h(this); ' + \_onkeyup);  }  }  });  }  },  //helpers  getQuestionDivByRID: function (rid) {  var \_me = this,  \_d = null,  \_dQuestion = $('div.question-container').filter(function (i, d) {  return $('[id="' + rid + '"]', $(d)).length > 0;  })[0];  if (typeof (\_dQuestion) !== "undefined") {  \_d = \_dQuestion;  }  return \_d;  },  // INITIALIZING FUNCTIONS  init: function () {  var \_me = this;  //wait for DOM readiness  $(function () {  //cal functions here  \_me.bindSkipPatterns();  \_me.checkSkipPatterns();  \_me.bindClearAbove();  \_me.applyMask();  });  }  }; |