

Using Accordion Templates for Expandable Rosters – HRS Experiences

Helena Stolyarova and Jason Ostergren, University of Michigan Institute for Social Research

Health and Retirement Study (HRS) at the University of Michigan Institute for Social Research is a longitudinal study that originated in 1992, switched to conducting interviews using Blaise 4 in 2002, and Blaise 5 in 2018. One of the challenges faced by HRS across this long period of Blaise development has been how to handle editable tables (we often refer to these as “rosters”) of people, such as children of respondents, or things, such as pensions. This challenge has intensified with the adoption of Blaise 5 and the corresponding move into browser-based self-administered interviewing over the internet. This paper will discuss the history of HRS rosters, including the various design goals and specifications involved, and it will also discuss the specific implementation, using “accordion” templates, of various rosters in an auxiliary instrument currently under development called Life History Mail Survey (LHMS). The latter is a longstanding mail supplement to the HRS, which is going to include an internet version of the survey for the first time this year. Overall, HRS has been looking for ways to make rosters that can dynamically expand (add more children, pensions, etc. with a large capacity), that can be preloaded, that work well on both desktop and phone browsers and form factors, and that offer interactive navigation between elements and summary information when not on the same page. For LHMS specifically, a new approach was tried which attempts to combine all elements and summary information into a single page by use of “accordion” templates. Details of this endeavor, including successful implementation of the basic design, as well as the many problems and limitations encountered will be discussed. In many ways, this new approach looks promising, but a number of obstacles will need to be overcome to ensure its future viability.

A Brief History of HRS Rosters

The first attempts to make rosters for HRS in Blaise started in 2001. At that point we were moving from another computerized survey software that our interviewers were already familiar with into Blaise 4 and the expectation was that our rosters would follow a similar grid format. In HRS, rosters are essentially lists of people or things connected to the Respondent (children, household members, siblings, pensions). In other words, these are arrayed collections of data, with each item in the array being a distinct identifiable “object” such as a person or pension, which tends to be intuitively perceived like a row in a spreadsheet table. Importantly in HRS, these lists include more than just a simple name or other identifying data. They tend to include what might be considered follow-up questions right in the roster itself. This small battery of questions is then further built upon by later sets of follow-ups or other questions based off the lists from the rosters. These rosters also tend to have complicated logic, not only for the flow within the individual roster row representing a person or pension, but also between rows and among the whole list. This latter type of logic contributed to the high complexity of a few rosters, such as that for gathering a list of children of the Respondent. Lastly, these rosters in the standard HRS have to account for preload, though crucially this is not a requirement in the LHMS study which will be described in the later part of this paper. Preload adds a massive additional dose of complexity, as it governs much of the roster logic with preloaded things being handled differently than new things, but all displayed in the same roster. Furthermore, HRS survey design mandates different handling for the interview of the spouse or partner respondent in the same roster when it is their turn. Needless to say, all this took a long time to iron out initially, with revisions every biennial wave to improve flow, data, interviewer experience, and, of course, to fix bugs that cropped up owing to the complicated design. Nonetheless, the Blaise 4 Table feature worked well for HRS for the approximately decade-and-a-half period during which we used Blaise 4 for data collection.

To give an idea of what this design entailed, and how it was experienced by interviewers, a screenshot is provided below (Figure 1) showing some of the things described above. An attempt will be made here to briefly describe key parts of this design and interface in slightly more detail. While this example shows a roster of children, each other type of roster came with its own set of unique complications, such as a requirement to add or remove people in the household and sibling roster, based on residency reporting elsewhere, or the need to merge job data with pension data on the fly in the pension roster.

Figure 1. Blaise 4 roster layout

	FIRST NAME	REL	SP	SEX	RES	MO	YR	DUP	SAM	WH?	MAR	PRT	NEW SP FIRST NAME	SP SEX	SP RES	COMMENT
	KEVIN	3	3	1	1						1		new spouse	2	1	
	KAREN	6	6	2	1						5	5				
1	NEW CHILD	3	3	1	1	1	2000				5	5				

HRS (that is, the main study) did not begin to conduct self-interviews of respondents until switching to Blaise 5 in 2018, so the Blaise 4 design of rosters seen above at least lacked that complicating factor – only trained interviewers would need to navigate these screens. Above, we see two preloaded children in the first two rows of the table pane who would have been reported in prior waves of HRS, one of whom has married a new spouse (named “new spouse” here for convenience). A new child is reported in the third row, and, since it is the first report of this child, we collect a birthdate, illustrating how the set of questions asked is conditional on various factors. In the last row, another new child is being added, and the roster can expand up to forty children in total. For a very long time, new spouses were added as a new row (as seen above), which drove up the complexity between rows a lot, but later HRS moved to expand the child row to collect new spouse information. However, HRS did retain, through its entire run with Blaise 4, preloaded spouses on their own line (not shown in the above example), with their own conditional set of questions.

One perennial problem with this design and the way it interacted with “second Rs” (the HRS term for whichever respondent in a household did their survey last) is that when the roster was populated, much depended on the codes for relationship to R (respondent) and to the spouse/partner (shown in the “REL” and “SP” columns in the screenshot above, utilizing a few dozen possible codes). If these codes

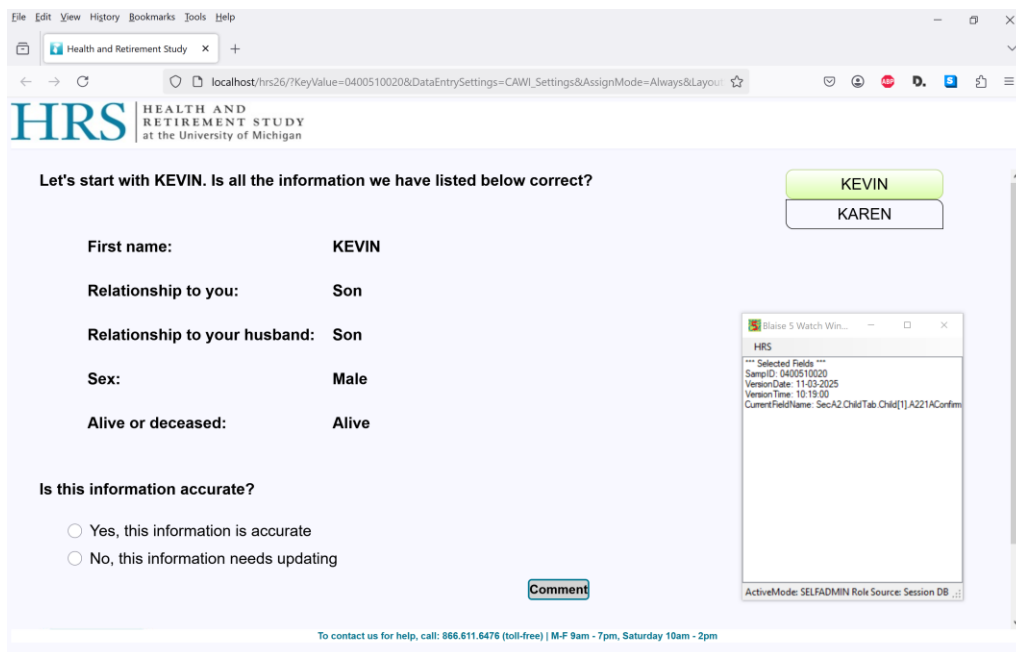
were not in at least general agreement, it would cause major headaches on this table. HRS interviewers were trained to be able to move the cursor back and change things if needed, but despite increasing efforts to limit, through flow logic and wording, opportunities for error, interviewers occasionally made mistakes, and respondents sometimes report relationships that seem to defy classification.

One attempt to fix mismatched relationships involved having a “reconciliation grid” preceding the child table which would show a summary of relationships entered the prior wave and allow them to be corrected before the child roster was started, making it possible to populate the child roster correctly. However, while this feature did help to clean up many problems, it was eventually abandoned because it ramped up the complexity of the code even more, took more respondent time, and failed to solve the problems almost as often as it helped. By the time HRS began design work in Blaise 5, we were primed to look for new approaches, especially since the main purpose of adopting Blaise 5 was to begin to facilitate web self-interviews.

HRS Rosters and the Transition to Blaise 5 with Support for Web Self Interviews

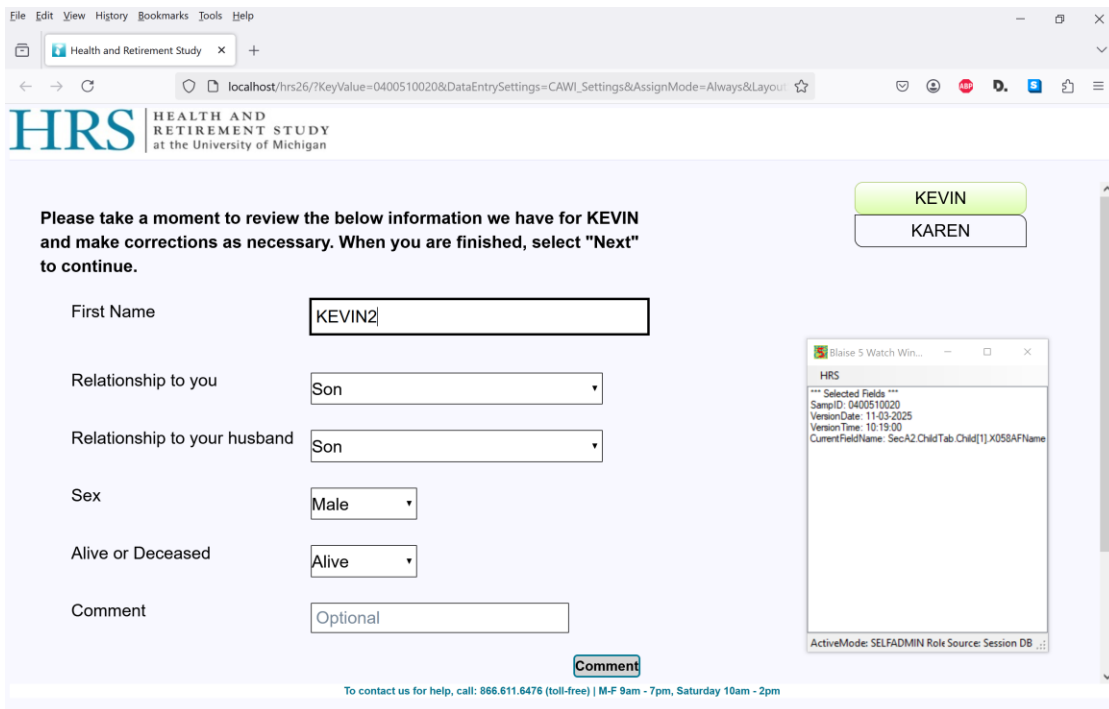
HRS decided to move to Blaise 5 when it became apparent that it would have to try to conduct its main interview on the web as a self-administered survey. At first this would be for a very small subset of the sample, but that number is growing each wave. This meant that these rosters needed to be much more user friendly. One key concept that emerged from preparatory work on the Blaise 5 transition was that, rather than having a large, potentially overwhelming number of fields confronting the respondent, each roster item should be broken down into key fields required for proper identification of the person or thing and an immediate follow-up loop containing fields that didn’t meet that criteria (still separate from other, more substantial follow-ups later in the instrument). Additionally, it became apparent that we needed to give respondents every opportunity to correct any mistakes in this critical identification data, so a summary screen would need to be provided between these two parts which allowed for corrections.

Figure 2. Web version preloaded child roster



The screenshot above (Figure 2) shows a preloaded child. A number of changes can be seen from the old Blaise 4 design, particularly driven by the need for self-interview ease of use. The most common situation facing a respondent in a roster like this will be that the list of items (identifiers for people or things) has not changed, even if the status of those items has changed (marital status, current residency, etc.). Because the roster is broken into two parts (identifiers and follow-ups) in Blaise 5, we now have a simple confirmation question for the identifier portion of previously reported items, children in this case. Rather than being asked about each piece of information separately as in the Blaise 4 design, a single answer of “yes, this information is accurate” will move on to the next item in the roster. We still provide the option to answer the questions separately if the respondent chooses the other option on the confirmation screen. In that case, through a bit of gated-flow trickery, the confirmation question is hidden and it is replaced by answerable fields for the identifiers as seen in the screenshot below (Figure 3). Note also that on the right side (or at the bottom in a mobile-device layout) there are buttons to navigate to other items in the roster to edit them as well.

Figure 3. Opened child roster



As in the Blaise 4 design, after the preloaded lines are handled, additional lines in the roster can be opened up with an “any other” question as shown in the screenshot below (Figure 4), which then opens up the same set of identifiers to be filled in, except that they are blank at first.

Figure 4. “Any other” question on roster

Do you or your husband have any other living children or step-children?

- Do not include children-in-law.
- Yes
- No

Please add information below for this child.

First Name:

Relationship to you:

Relationship to your husband:

Sex:

Comment:

Summary Table:

KEVIN2				
KAREN				

Blaise 5 Watch Win... HRS
 Selected Fields ***
 SampleID: 0400510020
 VersionDate: 11-03-2025
 VersionTime: 10:19:00
 CurrentFieldName: SecA2.ChildTab.Child[3].X058AFName
 ActiveMode: SELFADMIN Role Source: Session DB ...

To contact us for help, call: 866.611.6476 (toll-free) | M-F 9am - 7pm, Saturday 10am - 2pm

When the respondent finally answers “no” to the “any other” question, a summary screen is displayed (Figure 5), along with a “Summary Page” navigation button below the others.

Figure 5. Summary table

The table below is a summary of the information that you just entered or confirmed. Please review this information.

- If you see duplicate names on this list, please describe the problem using the comment button.

Child's Name	Relation to you	Relation to your husband	Sex	Alive or Deceased
KEVIN2	Son	Son	Male	Alive
KAREN	Daughter	Daughter	Female	Alive
john	Son	Son	Male	Alive

Children (edit)

KEVIN2

KAREN

john

[Add a child]

Summary Page

Blaise 5 Watch Win... HRS
 Selected Fields ***
 SampleID: 0400510020
 VersionDate: 11-03-2025
 VersionTime: 10:19:00
 CurrentFieldName: SecA2.ChildTab.A224_ChildSummary
 ActiveMode: SELFADMIN Role Source: Session DB ...

To contact us for help, call: 866.611.6476 (toll-free) | M-F 9am - 7pm, Saturday 10am - 2pm

It may be helpful at this point to note that the navigation buttons are a byproduct of using section index grouping templates for these rosters. The buttons appear on an IndexAreaList panel and their navigation functionality is an out-of-the-box feature of Blaise 5. Overall, this is not terribly different from tabs in parallel tabs, but HRS found this feature to work better for us in early testing and parallel tabs have some unwanted side effects for us. Nonetheless, like parallel tabs, the section index features are seemingly meant to span an entire instrument or at least are not intended to be re-used multiple times in the same one like we have done. HRS can get away with this because many of our rosters are gated, but if that were not the case, we would likely have problems where navigation buttons from previous grids appeared in later ones. That particular limitation has never been a problem for the HRS design, but we have always noticed poorer performance on these rosters compared to the rest of the instrument, likely linked to the various ways HRS has stretched Blaise functionality in the programming of these rosters.

Once the respondent presses “Next” on the summary screen, the aforementioned gate is closed (in most cases, such as on the child roster we have been using as an example) and it is no possible to return to these screens. In their place, a new summary screen, which, importantly, does not make use of the section index feature, is shown, should the respondent back up looking for the roster again. This serves both to remind them of their answers and to explain again why they can no longer change them, so that it does not seem to the respondent as though the answers have been lost.

Immediately following the summary screen, the respondent gets a conventional set of looped questions, with the remaining follow-ups for each child in order. For the child table in particular, we kept all questions about children in law for the follow-ups. Thus, preloaded children in law no longer even appear in the roster itself as they did in Blaise 4, reducing the respondent burden somewhat. This immediate set of follow-ups can be fairly lengthy depending on the type of roster, but the navigation and programming is simplified because the list of items to loop through has already been set in the roster itself. As mentioned at the beginning, this list of items will re-appear in later parts of the interview for use in additional follow-ups and codeframes relevant to the subject matter. There are a variety of other complexities under the hood, and there is a separate template and layout set for our interviewer-administered mode (though the underlying programming is the same), but this design and implementation has worked reasonably well for four waves of HRS now, since 2018. It has not, however, been simple to implement over this time and we are somewhat restricted in various ways from making big changes between waves of HRS due to the complexity of it. Thus, when we started to think about rosters in the supplemental LHMS study which is the main focus of this paper, we began looking for other approaches.

Layouts for Web Surveys: Design vs. Implementation

LHMS: Life History Mail Survey

Background: The Life History Mail Survey (LHMS) contains questions about residential history, education history, marriage and partnership, health conditions and other important childhood and family events. The LHMS is part of the Health and Retirement Study (HRS), which is funded under a cooperative agreement between the National Institute on Aging (NIA) and the Survey Research Center at the University of Michigan.

Paper and Pencil Design: A brochure, approximately 40 pages long, is mailed to respondents. The survey takes approximately 90 min to complete. Once done, respondents are asked to mail back the brochure. In 2024 HRS started an effort to convert LHMS into a Web Survey to be fielded in May of 2025 (Figure 6).

Figure 6. LHMS paper and pencil intro

ABOUT THIS SURVEY

This Life History Survey is a new part of the Health and Retirement Study. It will give us some information about important things that happened earlier in your life so that we understand better how you are doing now.

This survey is not meant to be a test of your memory. However, we would like you to try to be as accurate as possible. You may find it useful to consult your spouse, another family member, or some personal files, photos, or notes as you go through the questions.

We hope that you will find this survey interesting to complete. As always, your answers are extremely important to us. Please remember that your participation is voluntary and that you may skip over any questions that you would prefer not to answer.

Please return your completed Life History Survey in the pre-addressed postage paid envelope. If you have any questions, please feel free to call us at 1-866-611-6476.

HOW TO FILL IN THIS SURVEY

Please answer the questions by:

Marking a box like this:

Or writing in a box like this:

		2	5
--	--	---	---

Answer

Please use a #2 pencil or a blue/black ink ballpoint pen. DO NOT use a felt tip pen.

Sometimes you may find instructions telling you which questions to answer like this:

Q5

Yes (Continue to Q5)
 No → (Go to Question Q10 on page 5)

Some of the questions spread across two facing pages like this.

#	Start Year	Street (Number and Street)	City/Town	State (or Country)	Zip	Did you or your family own or rent this residence? (Check one)
1	1945	128 Apple Drive	Ann Arbor	MI	48104	<input checked="" type="checkbox"/> Own <input type="checkbox"/> Rent <input type="checkbox"/> Other <input type="checkbox"/> Don't know
2	1965	456 N 17th	Chicago	IL	60427	<input type="checkbox"/> Own <input checked="" type="checkbox"/> Rent <input type="checkbox"/> Other <input type="checkbox"/> Don't know
3						<input type="checkbox"/> Own <input type="checkbox"/> Rent <input type="checkbox"/> Other <input type="checkbox"/> Don't know
4						<input type="checkbox"/> Own <input type="checkbox"/> Rent <input type="checkbox"/> Other <input type="checkbox"/> Don't know

Please complete one row across both pages before moving to the next row.

2

9 7 4 8 1 3 0 2

Challenges: Many questions in the Paper-and-Pencil instrument are easily converted to a web layout (Figure 7).

Figure 7. Paper and pencil conversion of a table to web format

Q31 In high school, did you take classes or spend time to do the following:

a. Learn to play a musical instrument	<input type="checkbox"/> Yes	<input type="checkbox"/> No
b. Take singing lessons or sing in a chorus or choir	<input type="checkbox"/> Yes	<input type="checkbox"/> No
c. Learn woodwork or carpentry	<input type="checkbox"/> Yes	<input type="checkbox"/> No
d. Learn a craft (e.g., knitting, quilting, embroidery)	<input type="checkbox"/> Yes	<input type="checkbox"/> No
e. Learn ballet or dance	<input type="checkbox"/> Yes	<input type="checkbox"/> No
f. Learn to paint or draw or other art	<input type="checkbox"/> Yes	<input type="checkbox"/> No
g. Participate in math or science club	<input type="checkbox"/> Yes	<input type="checkbox"/> No
h. Learn drafting or technical drawing	<input type="checkbox"/> Yes	<input type="checkbox"/> No
i. Take vocational or trade classes (e.g., auto repair, HVAC)	<input type="checkbox"/> Yes	<input type="checkbox"/> No
j. Participate in theatre, drama, or debate club	<input type="checkbox"/> Yes	<input type="checkbox"/> No

HRS | HEALTH AND RETIREMENT STUDY

Life History Survey

In high school, did you take classes or spend time to do the following:

	Yes	No
Learn to play a musical instrument?	<input type="radio"/>	<input type="radio"/>
Take singing lessons or sing in a chorus or choir?	<input type="radio"/>	<input type="radio"/>
Learn woodwork or carpentry?	<input type="radio"/>	<input type="radio"/>
Learn a craft (e.g., knitting, quilting, embroidery)?	<input type="radio"/>	<input type="radio"/>
Learn ballet or dance?	<input type="radio"/>	<input type="radio"/>
Learn to paint or draw or other art?	<input type="radio"/>	<input type="radio"/>
Participate in math or science club?	<input type="radio"/>	<input type="radio"/>
Learn drafting or technical drawing?	<input type="radio"/>	<input type="radio"/>
Take vocational or trade classes (e.g., auto repair, HVAC)?	<input type="radio"/>	<input type="radio"/>
Participate in theatre, drama, or debate club?	<input type="radio"/>	<input type="radio"/>

Comment

Next ▶

◀ **Previous**

However, event history questions (residential, educational, marriage, employment and health histories) often span across two pages (Figure 8).

Figure 8. LHMS Paper and pencil residential history

Q5 In this table, please fill in as much information as you can about **all the places that you have lived for one year or more** from when you were born until **now**.
To begin, please enter the year of your birth and the place where you lived when you were born. Beginning in #2, write the **next** place where you lived for a year or more, and so on. If you can't remember the exact year(s), please estimate the year to the best of your ability. If you lived outside the U.S., write the country name instead of the state. **Use one line for each new place (see example p. 2).**

TABLE COLUMNS SPAN ACROSS BOTH PAGES. ►

#	Start Year	Street (Number and Street)	City/Town	State (or Country)
1	<input type="text"/>			
2	<input type="text"/>			
3	<input type="text"/>			
4	<input type="text"/>			
5	<input type="text"/>			
6	<input type="text"/>			
7	<input type="text"/>			

◀ TABLE COLUMNS SPAN ACROSS BOTH PAGES.

#	Zip Code	Did you or your family own or rent this residence? [Check one box]
1	<input type="text"/>	<input type="checkbox"/> Own <input type="checkbox"/> Rent <input type="checkbox"/> Other <input type="checkbox"/> Don't know
2	<input type="text"/>	<input type="checkbox"/> Own <input type="checkbox"/> Rent <input type="checkbox"/> Other <input type="checkbox"/> Don't know
3	<input type="text"/>	<input type="checkbox"/> Own <input type="checkbox"/> Rent <input type="checkbox"/> Other <input type="checkbox"/> Don't know
4	<input type="text"/>	<input type="checkbox"/> Own <input type="checkbox"/> Rent <input type="checkbox"/> Other <input type="checkbox"/> Don't know
5	<input type="text"/>	<input type="checkbox"/> Own <input type="checkbox"/> Rent <input type="checkbox"/> Other <input type="checkbox"/> Don't know
6	<input type="text"/>	<input type="checkbox"/> Own <input type="checkbox"/> Rent <input type="checkbox"/> Other <input type="checkbox"/> Don't know
7	<input type="text"/>	<input type="checkbox"/> Own <input type="checkbox"/> Rent <input type="checkbox"/> Other <input type="checkbox"/> Don't know

Why not use a table for the event history rosters?

1. Poor User Experience on Small Screens

- Tables do not scale well on mobile devices, often requiring users to scroll horizontally or zoom in/out to see content
- Touch interactions with small table cells can be frustrating, leading to mistakes

2. Visual Overload and Clutter

- Tables display all data at once, making it overwhelming—especially for long surveys with multiple sections
- Respondents may feel fatigued when faced with a large grid of empty or unnecessary fields

3. Fixed Structure, Limited Flexibility

- Traditional tables often have a preset number of rows, requiring extra fields that may not be needed
- Users cannot easily add or remove rows dynamically based on their life events

Design Goals & Requirements

- **Expandable Rows:** Users can expand or collapse rows to improve readability and navigation
- **Growing Tables:** Allows respondents to dynamically add or remove rows based on their life events (e.g., marriages, education, employment)
- **Responsive Layout:** Seamless transition between PC and mobile views for an optimized user experience
- **Summary View When Collapsed:** Displays key information in a concise format, helping respondents recall past life events

Design Advantages

- **Enhanced User Experience:** Reduces visual clutter by displaying only necessary information at a time
- **Mobile-Friendly:** Eliminates the awkwardness of large tables on small screens, making navigation smoother
- **Efficient Data Entry:** Users can add only relevant entries instead of dealing with a fixed number of rows
- **Memory Jogging Effect:** Research indicates that summarizing information in a collapsed state helps respondents recall details more accurately

Implementation Approach: EAR (Expandable Accordion Rosters)

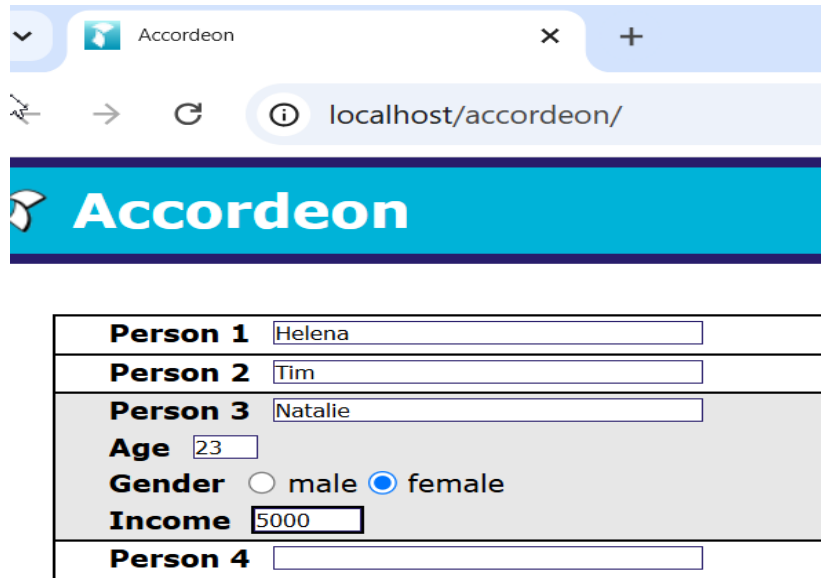
- **Accordion Roster:** Uses collapsible sections (rows) to present information efficiently while also serving as a summary screen
- **Dynamic Row Management:** Provides the ability to insert or delete rows as needed
- **Adaptive UI Design:** Ensures usability across different devices by switching between layouts appropriately
- **Automatic Summary Generation:** When collapsed, the accordion serves as a summary screen and displays key details from expanded sections to aid memory recall

EARs Development Process

We went over the Blaise sample projects provided in the folders ‘C:\Documents\Samples\Specific Features’ and ‘C:\Documents\Samples\Specific Features\Table Layout\Accordion\Accordion.bsol’.

We found two projects that when combined helped us address the design requirements described above.

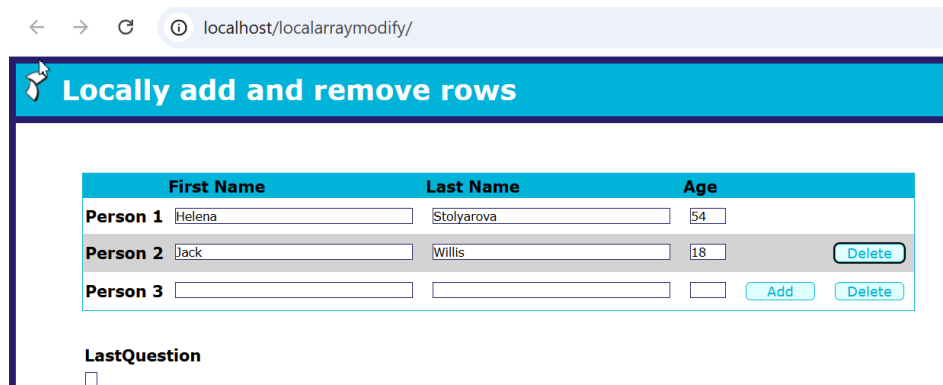
Figure 9. Blaise sample project 1



The screenshot shows a web browser window with the title 'Accordion' and the URL 'localhost/accordion/'. The page features a blue header with the word 'Accordion' and a white icon. Below the header is a form with four rows, each representing a person. The first three rows are expanded, showing input fields for name, age, gender, and income. The fourth row is collapsed, showing only the name input field.

Person 1	<input type="text" value="Helena"/>
Person 2	<input type="text" value="Tim"/>
Person 3	<input type="text" value="Natalie"/>
Age	<input type="text" value="23"/>
Gender	<input type="radio"/> male <input checked="" type="radio"/> female
Income	<input type="text" value="5000"/>
Person 4	<input type="text"/>

Figure 10. Blaise sample project 2



The screenshot shows a web browser window with the title 'Locally add and remove rows' and the URL 'localhost/localarraymodify/'. The page features a blue header with the text 'Locally add and remove rows' and a white icon. Below the header is a table with three columns: 'First Name', 'Last Name', and 'Age'. The table has three rows. The first two rows are expanded, showing input fields for first name, last name, and age. The third row is collapsed, showing only the first name input field. Below the table are two buttons: 'Add' and 'Delete'. Below the buttons is a section titled 'LastQuestion' with a checkbox.

	First Name	Last Name	Age	
Person 1	<input type="text" value="Helena"/>	<input type="text" value="Stolyarova"/>	<input type="text" value="54"/>	
Person 2	<input type="text" value="Jack"/>	<input type="text" value="Willis"/>	<input type="text" value="18"/>	<input type="button" value="Delete"/>
Person 3	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Add"/> <input type="button" value="Delete"/>

LastQuestion

Challenges

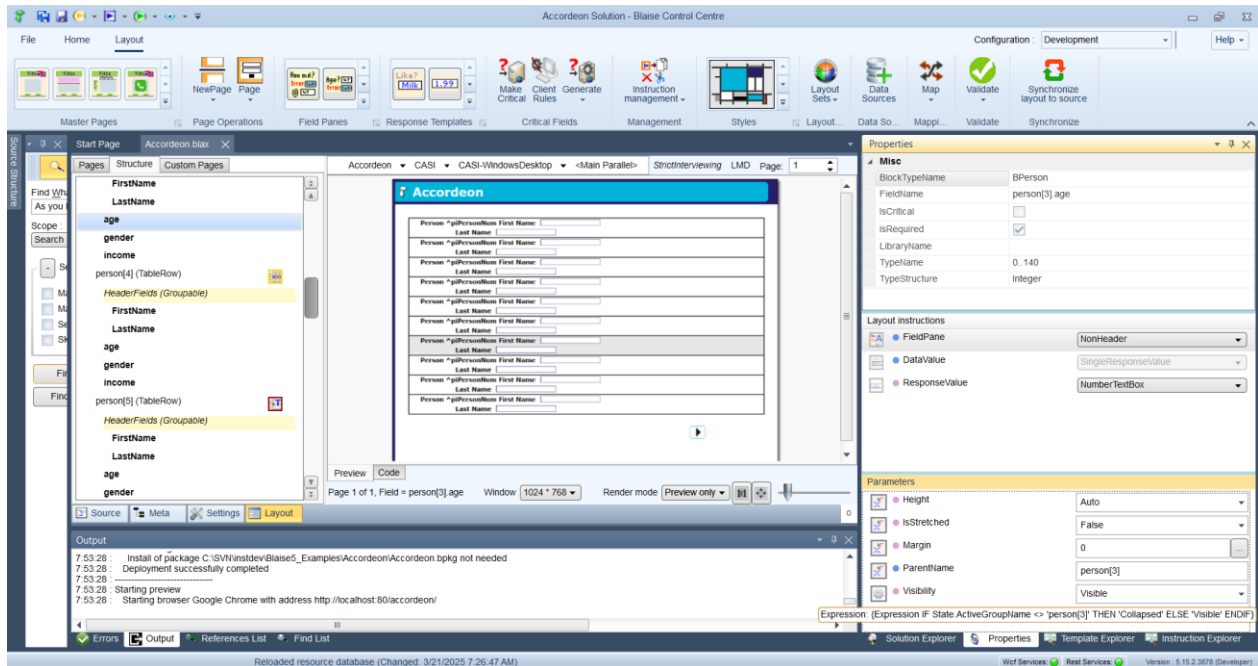
In the accordion example in the Blaise Sample folder, the inner accordion rows collapse/expand based on the Visibility Condition using parent name:

```

IF State.ActiveGroupName <> ParentName THEN
  'Collapsed'
ELSE
  'Visible'
ENDIF

```

Figure 11. Accordion example in the Blaise Sample folder

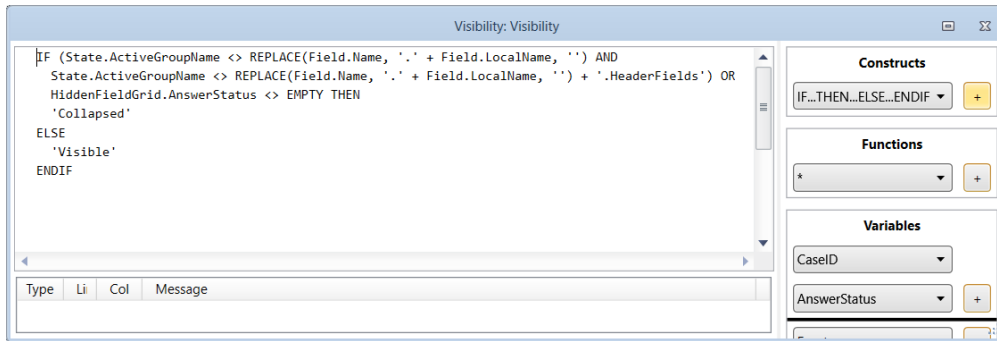


Problems and solutions

- Only four rows had *Parent Name* specified: person[1] – person[4]
- When we added more rows, *ParentName* was empty and the visibility condition did not work
- We needed more “generic” visibility condition, that also allows flexibility in naming rows: person[i], college[i], marriage[i], employment[i], etc.
- We needed more than one field in the accordion header rows. We created a group *HeaderFields*. This allowed us to put all the fields that we want in the top(header) row and use the name *HeaderFields* in visibility condition
- To find out the name of the group that is currently active (hence, needs accordion fold expanded), we created some “fake” fields on the page so that we can look at the *State.ActiveGroupName* variable. It would be really nice to have a built-in debugger tool that would allow users to observe the values of State variable

Our condition for the visibility of the accordion fold (Figure 12):

Figure 12. Condition for the visibility of the accordion fold



Field Name in the accordion fold:

ResidentialHistory.ResidentialTableRoster.RESIDENTIAL_ROASTER.residence[3].LH22_State

Remove Local Name using Replace function

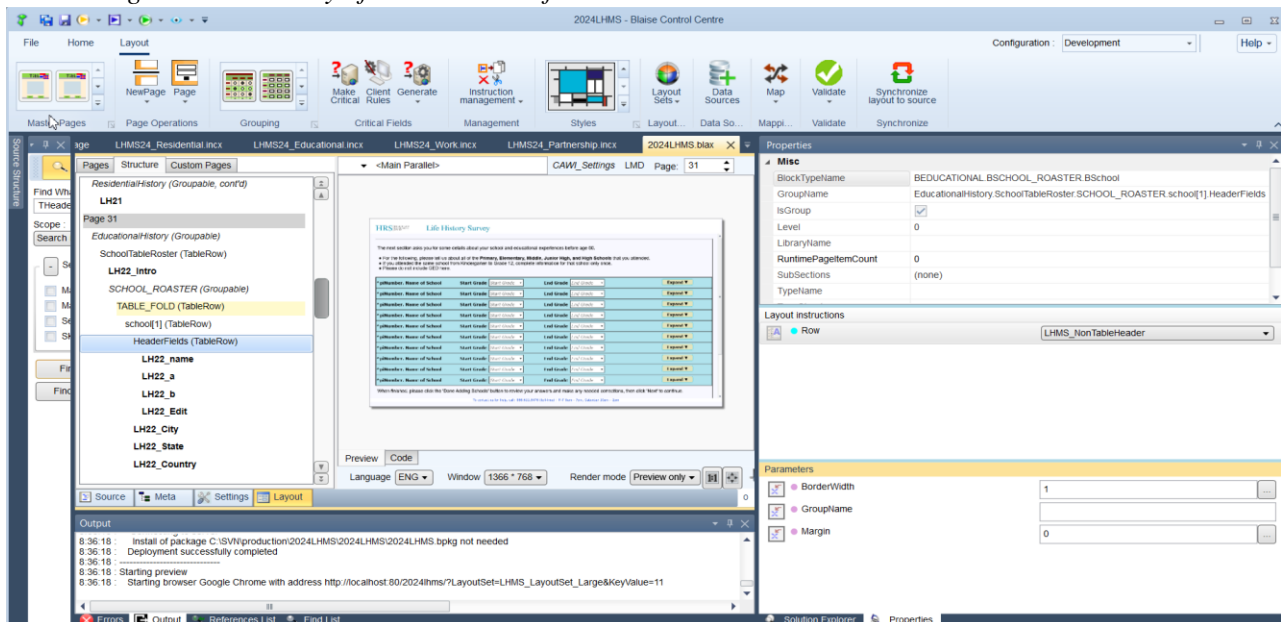
ResidentialHistory.ResidentialTableRoster.RESIDENTIAL_ROASTER.residence[3]

Add group name HeaderFields to get the name of the active parent group to be used in Visibility Conditions:

ResidentialHistory.ResidentialTableRoster.RESIDENTIAL_ROASTER.residence[3].HeaderFields

Now, when the group is active, the visibility condition of the accordion fold is set to 'visible' and the accordion row opens. This condition can be used for any roster as long as the header fields are defined in HeaderFields group (Figure 13).

Figure 13. Visibility of the accordion fold



EARs main features

Figure 14. Accordion fold on LHMS Web

FRS Life History Survey

- For the following, please fill in as much information as you can about all the places that you have lived for one year or more from when you were born until now.
- To begin, please enter the year of your birth and information about the place where you lived when you were born.
- Then, add the next place where you lived for a year or more, and so on.
- If you can't remember the exact year(s), please estimate the year to the best of your ability.

1. Start Year	1970	Street Address	123 Palm Street	City/Town	Moscow	Collapse ▲
State Wyoming						
Country (if outside of US)						
Zip Code 12345						
Did you or your family own or rent this residence? <input type="radio"/> Own <input type="radio"/> Rent <input checked="" type="radio"/> Other <input type="radio"/> Don't know						
Add next residence Delete residence						
2. Start Year	2002	Street Address	345 Deer Run Avenue	City/Town	Carson	Expand ▼
3. Start Year	2003	Street Address	3789 Curlew Lane	City/Town	Ann Arbor	Expand ▼

When finished, please click the 'Done Adding Residences' button to review your answers and make any needed corrections, then click 'Next' to continue.

Done Adding Residences

Comment

- **Add next residence** / **Delete residence** buttons functionality was implemented using the logic provided in the Sample Folder solution:

Specific Features\Table Layout\ExpandableTable2\LocalArrayModify.bsol

- **Expand ▼** / **Collapse ▲** functionality was implemented by mapping .blah level variable *HiddenFieldGrid* and using its value in the Visibility Condition. When “Expand” is clicked, a value is assigned to the mapped field *HiddenFieldGrid*. When “Collapse” is clicked, the value of the variable is set to EMPTY and the row collapses
- A parallel layout set was designed for mobile devices and screens with smaller resolutions (Figure 15). We used the same approach but arranged the header rows of the rosters vertically. The layout will switch automatically when a mobile phone is detected or when the PC screen is scrunched

Figure 15. Mobile device layout

Checks and Signals

Figure 16. Checks and Signals on accordion fold

One of the requirements was to add check/signals to the rosters. In our design, a user presented with multiple soft checks on the same page should be able to review and correct the entries. However, in HRS surveys, the hard checks are discouraged and web respondents are almost always allowed to proceed, even if the entries are obviously erroneous. Since multiple errors can be thrown on the grid, we added the logic to the “NEXT” button’s OnClick event that attempts to suppress all signals that are active on the current page.

However, we discovered an issue with this approach. In general, the problem is when multiple errors are made on the page, it is possible to proceed to the Next Page without notifying the user of all the errors. We are currently working with CBS to solve the issue.

Conclusion

The transition from traditional table-based roster designs to an expandable accordion template represents a significant step forward in improving user experience for web surveys. By addressing key challenges such as mobile responsiveness, visual clutter, and dynamic row management, the Expandable Accordion Rosters (EARs) approach has shown great promise in streamlining data collection while maintaining accuracy and usability.

Our implementation in Blaise 5 leveraged a combination of sample features and custom adaptations to overcome technical limitations. Despite the challenges (signals are not working properly), the accordion design offers notable advantages, including an intuitive interface, reduced cognitive load, and improved respondent engagement.

Moving forward, further refinements will be needed to enhance usability, optimize performance, and integrate debugging tools to facilitate development. Nevertheless, our experience with EARs in the Life History Mail Survey demonstrates the viability of this approach for future web-based survey instruments. Expanding its application in the broader HRS study and other longitudinal surveys could further validate its effectiveness and provide additional insights into best practices for dynamic roster management in web-based data collection.