Tasks

CSCC01

Abraham Tsang, Chengli Yang, Vijanthan Thiruchelvarajah, William Rutherford

10/23/2017

User Story 1: As professor Harris, I can CREATE a multiple-choice question with one expected answer.

- **Task 1-1:** Create (sqlite) database schema to hold questions. (3)
  Description: A multiple-choice question consists of a string containing a label for the question, a string for the text of the question, as well as a list of strings that holds candidate answers, with the first in the list corresponding to the correct answer. The question label needs to be unique and will be used as a primary key.
- **Task 1-2:** Create java function to insert question to database. (3) Depends on 1-1.
  Description: The function takes as parameters a string corresponding to the question label, a string for the question text, and an array of strings corresponding to multiple choice answers, and returns a string which indicates if the the question was successfully inserted, or an error message corresponding to why the question was not inserted.
- **Task 1-3:** Create GUI for professor to enter the question, choices, and answers. (6)
  Description: We will be building the UI using JFrames and we will be using WindowBuilder to assist us in creating this UI. We expect the result to support the creation of multiple choice questions of up to 6 choices, possibly inputted using input dialog boxes, and then display the question that was created on screen. Developer must test boundary cases to see that there are no nullpointer exceptions thrown such as when a user decides to press cancel on an input dialog box. They must also check that the label for the question and answer supports text wrapping.
- **Task 1-4:** Link GUI with java function. (2) Depends on 1-2, 1-3.
  Description: Once a question is successfully created, the program will invoke the API to insert this question into our sqlite database. The developer must then verify that the database is actually retrieving and storing the question when using our API.. Test on all boundary cases, including cancels, empty strings, and any malicious strings that try to invoke a query on the database.

User Story 2: As professor Li, I can SELECT a question to ADD to the assignment (there is currently only one assignment, the ability to create additional assignments is a subsequent user story).

- **Task 2-1:** Create (sqlite) database schema to hold assigned questions. (1)
  Description: The question labels (the primary key for questions in the questions table) will be recorded for assigned questions.
- **Task 2-2:** Create java function to insert question assignment to database. (1) Depends on 2-1.
  Description: This function takes the question label as a parameter and returns a string indicating successful assignment or an error message.
- **Task 2-3:** Create java function to retrieve available questions from database. (1) Depends on 1-2.
  Description: This function takes no parameters and returns data from all available unassigned questions, including their label, question text, and possible answers.
- **Task 2-4:** Create GUI to list available questions for professor to select and add one to assignment. (3)
  Description: We will build this UI as a JFrame and use WindowBuilder to assist us in creating this UI. The JFrame will be showing just a fixed sample of available questions in a listbox or listview and a user should be able to select one to add to the assignment. Developer must test to verify that all questions give expected output when selected to add. They must also check that the questions can extend past the max size of the displayed list with a scrolling mechanism.
- **Task 2-5:** Link GUI with java function to insert question assignment to database. (2) Depends on 2-2, 2-3, 2-4

Description: The fixed sample questions in 2-4 should be replaced with the actual list of questions from invoking our function from 2-3 to retrieve questions from the database. Verify that all questions are displayed. Upon selecting a question and adding such question, the developer must verify that the database is now storing this question assigned to the assignment.

User Story 3: As professor Li, I can VIEW the questions in the assignment.
- **Task 3-1:** Create java function to retrieve assigned questions from the database. (1) Depends on 2-2.
  Description: This function takes no parameters and returns data from all assigned questions, including their label, question text, and possible answers.
- **Task 3-2:** Create GUI to show assigned questions for the assignment (1)
  Description: We will build this UI as a JFrame and use WindowBuilder to assist us in creating this UI. This task is very similar to 2-4, basically a subset of it, in which it will show a fixed sample of questions in a listbox or listview. The list should be able to be scrolled through, so its' size is not limited to the size of the form window.
- **Task 3-3:** Link GUI with java function (1) Depends on 3-1, 3-2
  Description: The fixed sample questions in 3-2 should be replaced with an actual list of questions from invoking our function from 3-1 to retrieve questions from the database. Verify that all questions are displayed.

User Story 4: As student Noukovich, I can VIEW the questions in the assignment and ANSWER them.
- **Task 4-1:** Create java function to retrieve assigned questions from the database. (1) Depends on 2-5
  Description: This function takes no parameters and returns data from all assigned questions, including their label, question text, and possible answers. Similar to 3-1.
- **Task 4-2:** Create GUI to show assigned questions for the assignment AND allow entry of answers. (4)
  Description: Create this UI as a JFrame and use WindowBuilder to assist in creation. Similar to 3-2.
- **Task 4-3:** Link GUI with with java functions to retrieve assigned questions (1) Depends on 4-1, 4-2
  Description: Connect the GUI to the java function to allow the GUI to retrieve questions from the database and populate its fields.

User Story 5: As student Andrew, I can complete the assignment, and immediately after completing be given the grade I scored on the assignment.
- **Task 5-1:** Create java function to check provided answers for specific questions against expected answer. (2) Depends on 1-2, 2-2
  Description: This function takes the question labels and provided answers as parameters in a compound array and returns an array of strings which indicate if the answers were correct or the correct answer that was expected.
- **Task 5-2:** Create/modify GUI to display submitted answers and correctness. (1)
  Description: Again using JFrame/WindowBuilder, shows the student a list of questions with either "correct" or the expected answer for each question.
- **Task 5-3:** Link GUI from US4 to java function to check answers. (1) Depends on 4-2, 5-1
  Description: Connect the GUI to the java function to allow the inputted answers to be checked against the questions in the database.

- **Task 5-4:** Create logic to bring up GUI to display answers after checking answers. (1)
  Depends on 5-2, 5-3
  Description: Connect the java function to bring up the GUI for displaying submitted answers and correctness after checking provided answers against the questions in the database.