Mental Health and Wellness Screener
Software Design Specification

L. Vandecasteele (lv) N. Onofrei (no) A. Reichhold (ar) Q. Fetrow (qf) M. Gao (mg) -
2-14-2022 - v0.02

Table of Contents

1. SDS Revision History

Date	Author	Description
2/14/2022	lv	Created the initial document.
2/15/2022	no	Worked on System Overview and Software Architecture
2/15/2022	qf	Created Diagrams for use cases and overall Software Architecture
2/15/2022	mg	Added Text Describing Use Cases Models, Design Rationale
2/15/2022	lv	Slight error checking and grammatical changes
3/04/2022	qf	Updated all SDS modules and diagrams based on new functionality

2. System Overview

The Mental Health and Wellness Screener system is a software built with python that allows the user to take a mental health and wellness screener. The system presents questions, stores the data, and presents the data once the screener is finished.

3. Software Architecture

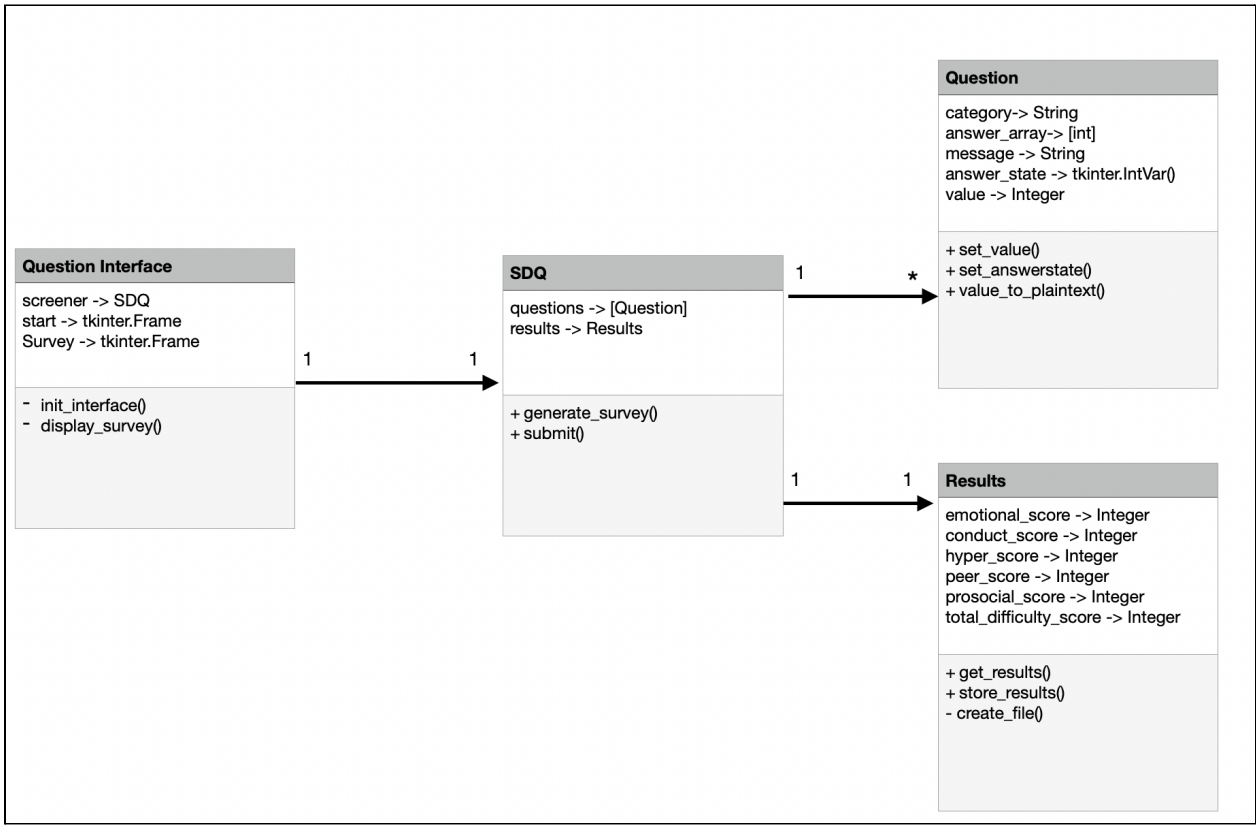


Figure 1: Software Architecture Model for Mental Health and Wellness Screener

Software Architecture - Modules

Question

The Question Module stores all information for each question including the text, how it will be scored, and the value submitted by the user. It is sent update commands by the Strengths and Difficulties Questionnaire module.

Strengths and Difficulties Questionnaire

The Strengths and Difficulties Questionnaire's module stores all the questions that are asked as part of the mental health and wellness screener, as well as an instance of the Results module that it calls methods on to score and store data. This module communicates with the Question Interface module so that the questions can be displayed.

Question Interface

The Question Interface Module allows the user to see/read/answer the question(s). This module holds various tkinter elements that enable user interaction, as well as an instance of the Strengths and Difficulties Questionnaire module, which it can call methods on depending on user interaction.

Results

The Results module will take all the questions that were answered and score them based on the standard Strengths and Difficulties Questionnaire scoring metrics. It will then store these results in an external file.

Design Rationale

Having all the questions in one module makes it easy to add and remove questions. It is then possible to keep track of the scores for each question to calculate the scores. Putting all the results in one module organizes the score and keeps the process of storing the data all in one module. The question interface deals with all the visual parts of the system software. This organizes the modules appropriately, allowing this module just to focus on the visualization part.

4. Software Modules

4.1. Question

The module's role and primary function.

The Question module is meant to store all question data, as well as the answer provided by the user. Questions are displayed by the **Question Interface** module, updated by the **Strengths and Difficulties Questionnaire** module, and graded by the **Results** module.

Interface Specification

Attributes:

- **category -> String:** The Category keeps track of which of the five question types (Emotional problems, conduct problems, hyperactivity, peer problems, and prosocial) the question is.
- **answer_array -> [int]:** The answer_array holds an array that represents how a question will be scored based on a value. For example, for a score_array of [2,1,0] and an answer_state of ID 0, the question value should be score_array[0] (= 2). This attribute is implemented because every question on the Strengths and Difficulties Questionnaire screener has options "Not True", "Somewhat True", and "Certainly True", but some questions can be scored differently.
- **message -> String:** Holds the text of the question.
- **answer_state -> tkinter.IntVal():** This attribute holds a variable tied to the question's radio buttons, and is representative of the current state of the answer.
- **value -> int:** Used to store the "difficulty score" for the question answered. A lower value indicates less of a problem in the question's category. For example for the question "I am restless, I cannot stay still for long" in the "hyperactivity" category, a high value would mean the student has more of an issue with hyperactivity.

Methods:

- **set_value():** Called by the **Strengths and Difficulties Questionnaire** module, sets **self.value** to be the integer value at **self.score_array[self.answer_state.get()]**. This value will be read by the **Results** module for score calculations.
- **set_answerstate(value-> tkinter.IntVal()):** Called by the **Question Interface**, ties the **answer_state** variable to the question's radio buttons shown on the display

- **value_to_plaintext() -> String:** Called by the results module to simplify the process of storing the students answers, returns the answer's integer value (0,1,2) to the plaintext version: ("Not True", "Somewhat True", "Certainly True").

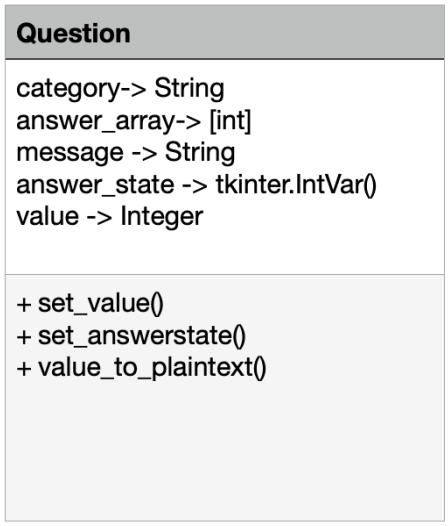


Figure 4.1.1: UML Class Diagram of Question Module

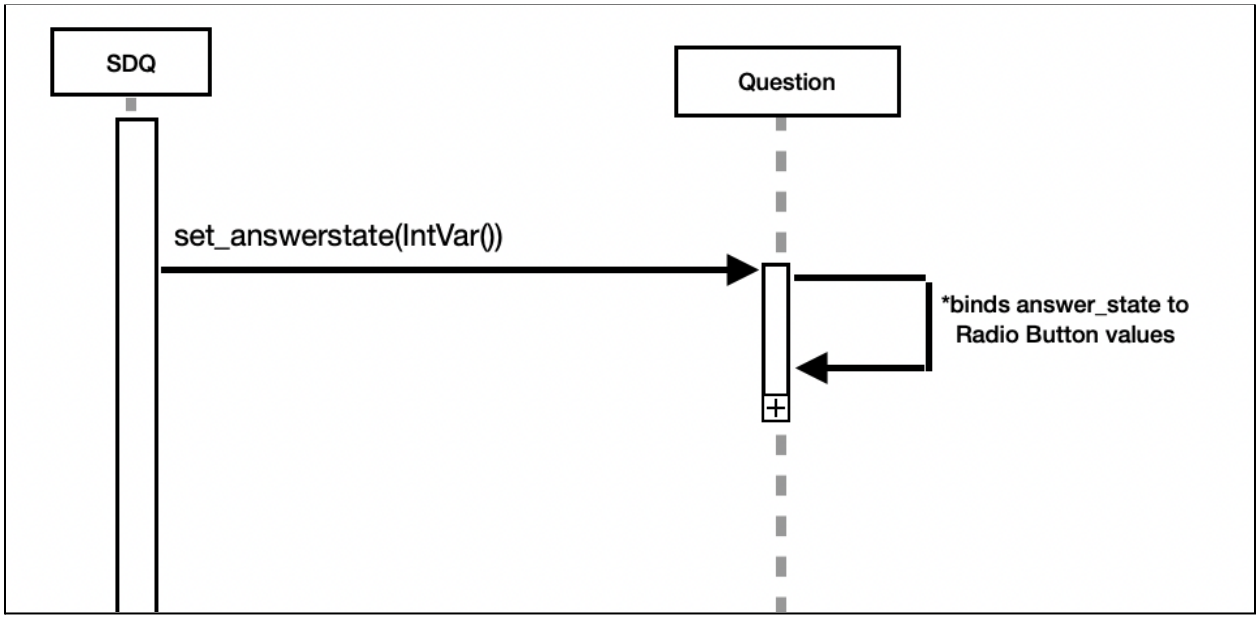


Figure 4.4.2: A UML Sequence Diagram of the Question module when interface is generated

Design Rationale:

This module is designed to hold all the question data and respective student answers such as category, score, text, etc. and places all relevant question information within a central spot. This allows modules such as the **Question Interface** and **Results Module** to quickly manipulate and access question information.

Because each question in the Strengths and Difficulties Questionnaire is tied to the same values "Not True", "Somewhat True", and "Certainly True", these answers are represented as integer values: [0,1,2] respectively. This can get confusing however, as some questions are scored differently. For example, the question "I usually do as I am told" would give a higher difficulty score if the student answers "Not True" whereas the question "I am often accused of lying or cheating" which would give a lower difficulty score for the same answer. The **answer_array** serves as a translation table for how to score these questions, while maintaining the constant [0,1,2] values tied to each question.

Some alternatives that were considered were storing the answer values in plaintext, e.q **value=**"Not True". Question values would then be set by assigning variables to the string values. This was decided against however as storing the answers in integer representations simplifies the process of translating the answers into difficulty scores.

4.2. Strengths and Difficulties Questionnaire

The module's role and primary function.

The Strengths and Difficulties Questionnaire holds instances of many **Question** objects, and is responsible for generating questions, and calling methods to update answer values of **Question** modules and score them.

Interface Specification

Attributes:

- **questions -> [Question]:** An array of **Question** modules that represent the survey.
- **results -> Results:** An instance of the Results module used to calculate and store results

Methods:

- **generate_survey():** Called on startup, reads and parses a file containing all questions on the standard Strengths and Difficulties Questionnaire screener, the category of the question, and the way the question is scored. The module then creates an array of questions to represent the survey. This array is manipulated by the Question Interface.

- **submit():** Bound to the submit button on the Display Module, calls get_results() and store_results() on the **Results** module.

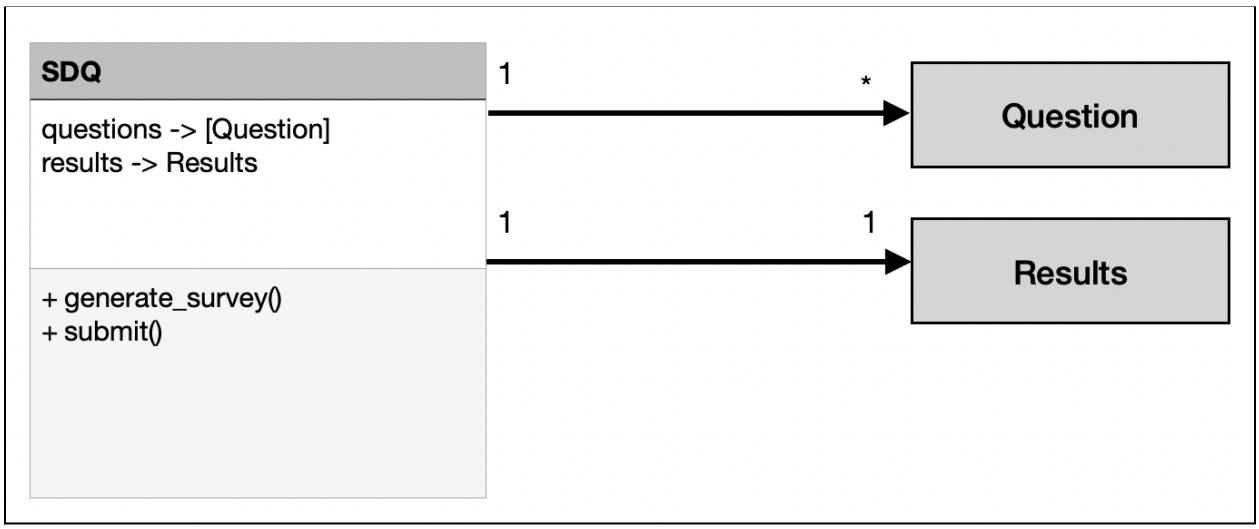
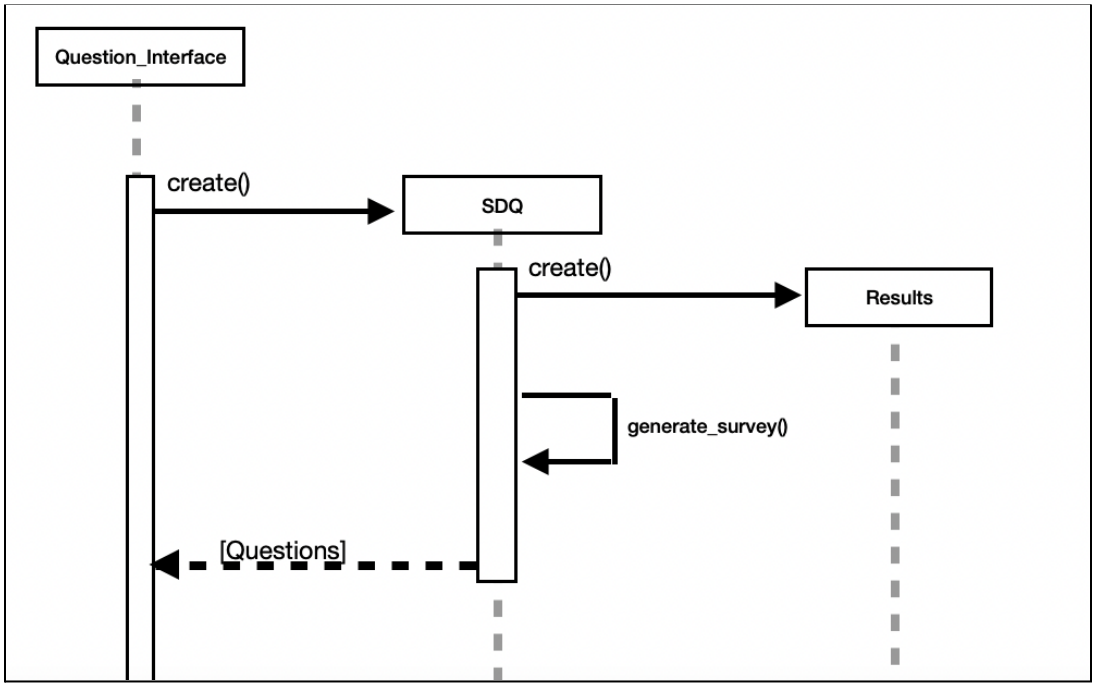


Figure 4.2.1: UML Class Diagram of the Strengths and Difficulties Questionnaire Module



cp

Figure 4.2.2: UML Sequence Diagram of Strengths and Difficulties Questionnaire when it is created by the Question Interface

Design Rationale:

The Strengths and Difficulties Questionnaire serves as a representation of the collection of questions in the screener, and serves to separate the user interaction from the stored values of questions and results. It was designed to be separate from the **question interface** so that the interface does not perform any file I/O.

Some alternatives that were considered were having the **Strengths and Difficulties Questionnaire** module hold an instance of the **Question Interface** module and update it every time a user answers a question. This type of controller-view interaction is more complicated however, as the **Question Interface** has many methods that need to work independently of the **Strengths and Difficulties Questionnaire.**

4.3. Results

The module's role and primary function.

The Results module calculates and stores the results for each category of questions, as well as the total question score. It is created and called upon by the **Strengths and Difficulties Questionnaire** module, and reads data from an array of **Question** modules to calculate the scores. It can then save these scores in an external file.

Interface Specification

Attributes:

- **emotional_score -> Int:** Integer to keep track of scores of questions in the "Emotional Problem Scale" category.
- **conduct_score -> Int:** Integer to keep track of scores of questions in the "Conduct Problem Scale" category.
- **hyper_score -> Int:** Integer to keep track of scores of questions in the "Hyperactivity Scale" category.
- **peer_score -> Int:** Integer to keep track of scores of questions in the "Peer Problem Scale" category.
- **prosocial_score -> Int:** Integer to keep track of scores of questions in the "Prosocial Scale" category.

- **total_difficulty_score -> Int:** Integer to keep track of the total difficulty score

Methods:

get_results([Question]): Called from the **Strengths and Difficulties Questionnaire** module, this method reads the values of each **Question** module in the sent array, and increments the integer attribute representing the **Question.category** and the **total_difficulty_score** by **Question.value**.

store_results(): Called from the **Strengths and Difficulties Questionnaire** module, stores the results of the questions to an external file.

create_file() -> File: Creates and returns the output file to be written to.

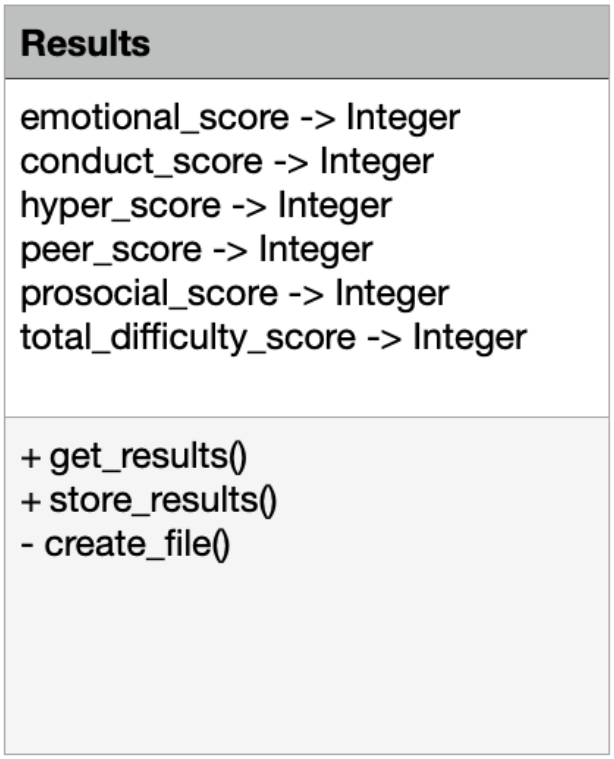


Figure 4.3.1: A UML Class Diagram of the Results Module

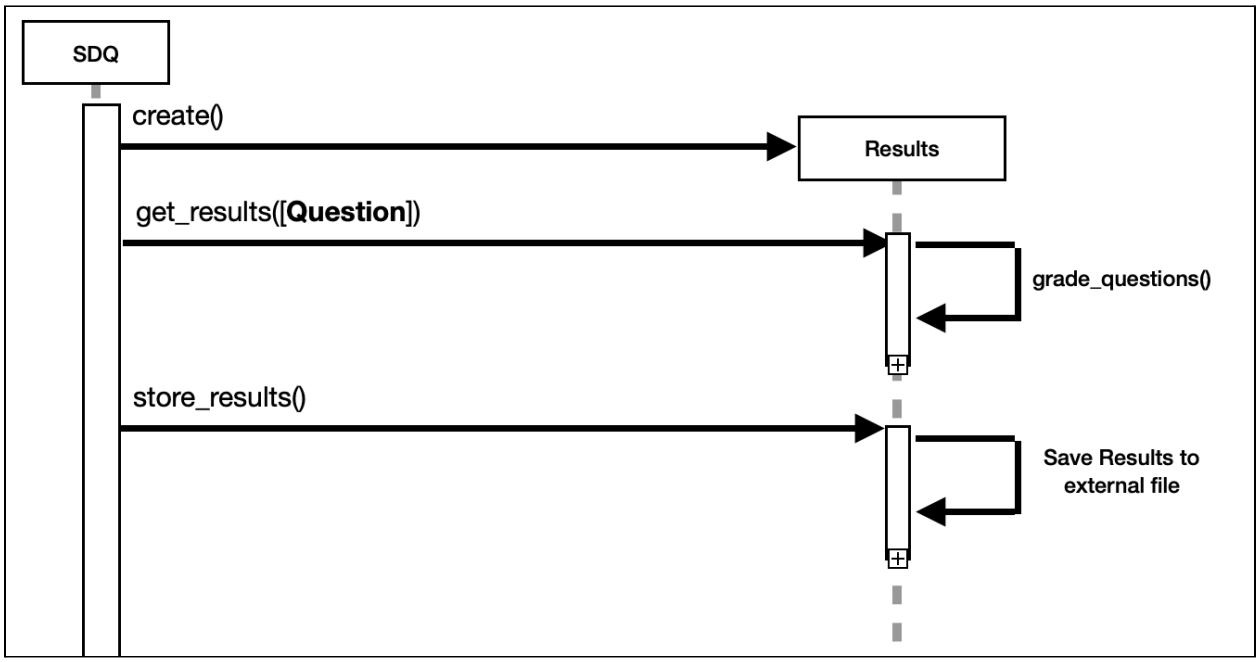


Figure 4.3.2: A UML Sequence Diagram of the Results Module when the user submits survey

Design Rationale:

This module allows all questions to be easily graded for mental health results. By having the results method write to an output file, it allows the student to easily send these results anonymously to the instructor. Some alternatives that were considered were creating a display to display the results to the student. This was decided against however, as it seems counterintuitive to display to students statistics about their own mental health and how they compare to other students.

4.4. Question Interface

The module's role and primary function.

The question interface is responsible for displaying the information and interactive elements to the user. It will hold various TKinter elements, including interactive buttons and text for each question, as well as an instance of the **Strengths and Difficulties Questionnaire** module that generates, stores, and scores the questions in the screener.

Interface Specification

Attributes:

- **screener -> SDQ:** An instance of the **Strengths and Difficulties Questionnaire** module to be called upon depending on user interactions.
- **start -> [TKinter Object]:** A representation of the interface created from the **init_survey()** method.
- **survey-> [TKinter Object]:** A representation of the interface created from the **display_survey()** method.
- **final-> [TKinter Object]:** A representation of the interface created from the **final_window() method.**

Methods:

- **init_interface():** This method generates the "Start Screen" that informs the student of the purpose of the screener and instructions. This interface holds a button to begin the screener, which calls **display_survey()** and draws
- **display_survey():** This method is called once the user presses the "Begin Survey" button. It calls **SDQ.generate_survey()** and fills the **question_interface** array with objects created for each **Question** in the **Strengths and Difficulties Questionnaire module**. The TKinter objects in the **question_interface** array are displayed to the user, as well as radio buttons that represent the answers of each question. Each question has three radio buttons for "Not True", "Somewhat True", and "Certainly True" answer values. These buttons are bound to a variable in the **Question module** through the **Question.set_answerstate(variable)** method. It also generates a **Submit** button that is bound to the **SDQ.submit()** method.
- **final_window():** This method is bound to the submit button in the **display_survey()** method. It generates the screen displayed to the user once they have finished the survey, which provides them with two button options. The first button, entitled "Take Again", calls **init_interface()**, restarting the program. The second button, entitled "Finish", destroys the widget, ending the program.

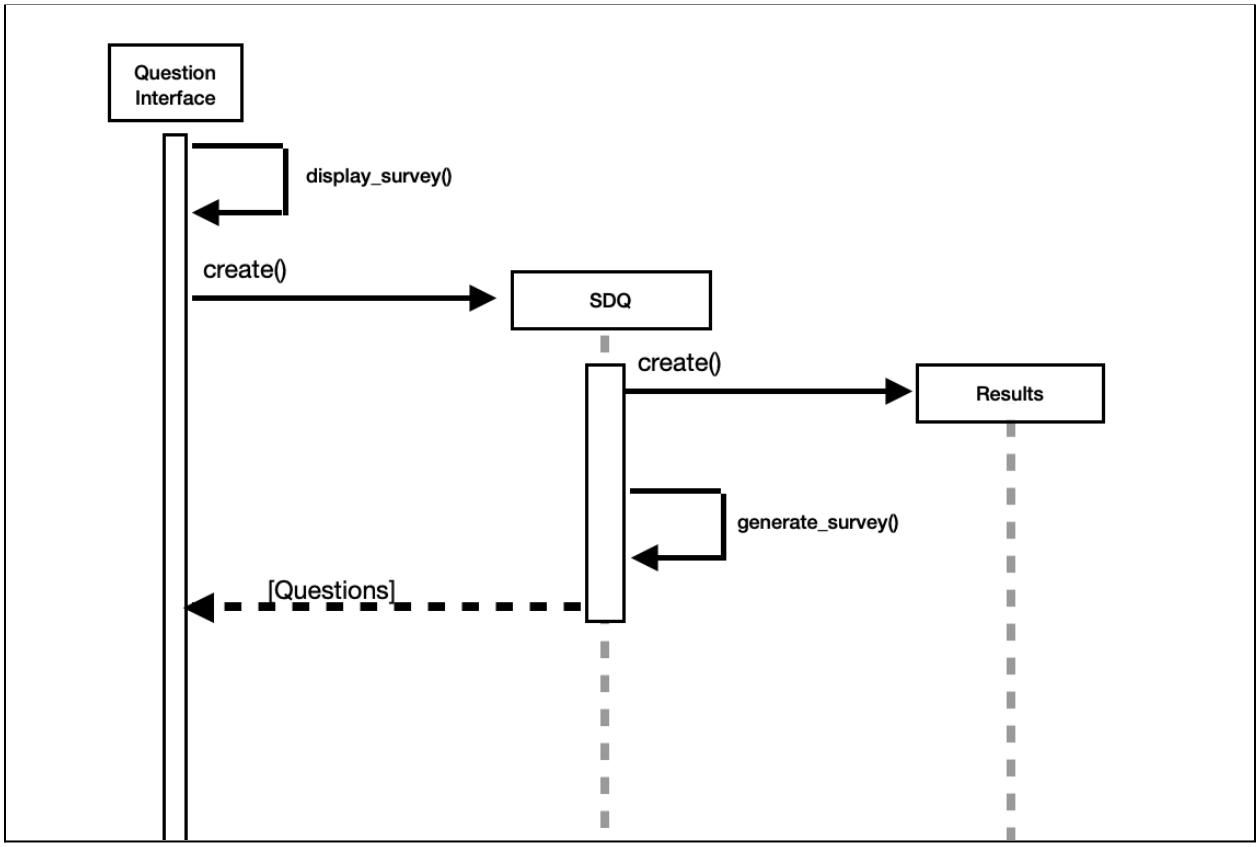


Figure 4.4.1: A UML Sequence Diagram of the Question Interface when it generates the survey.

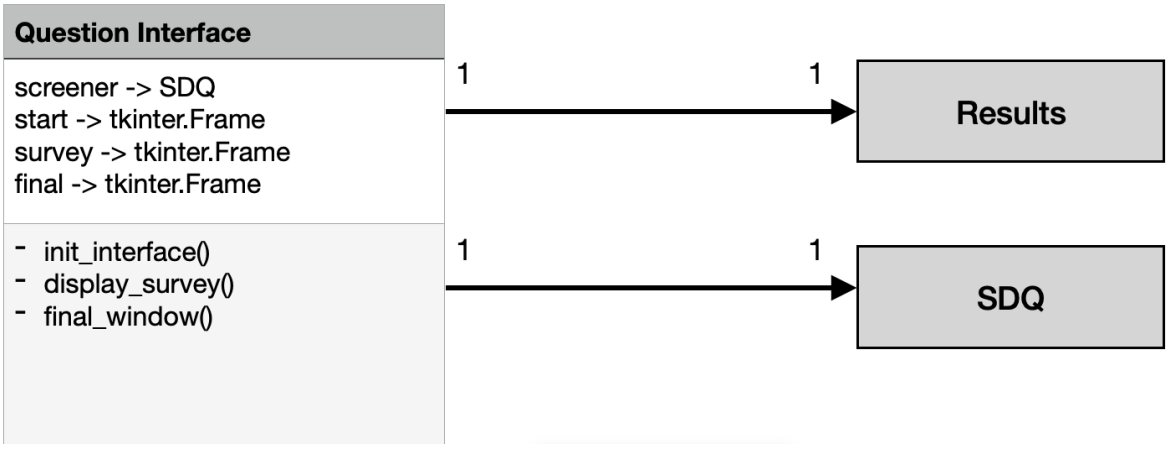


Figure 4.4.2: A UML Class Diagram of the Question Interface module.

Design Rationale:

This module allows for a visual representation of the student's questionnaire answers by providing an interactive GUI for the individual in question. Without this module, the student results would be presented as a text file, which is dull and static. The GUI is what the user sees, which is a critical and necessary component for user interaction.

5. Dynamic Models of Operational Scenarios (Use Cases)

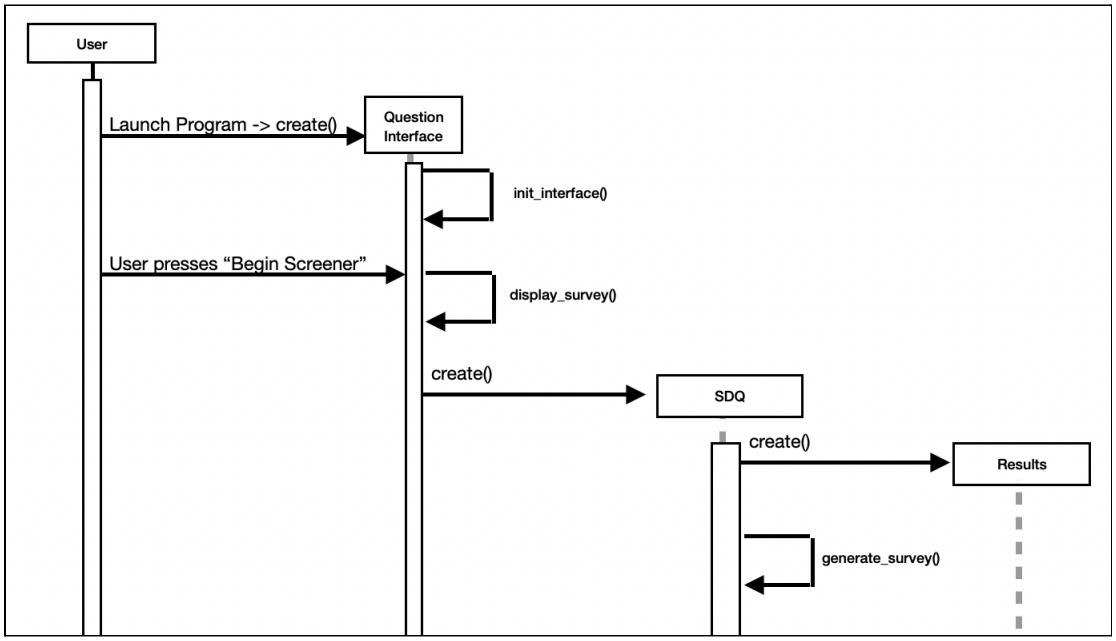


Figure 5.1: A UML Sequence Diagram of when the program is started

Figure 5.1 demonstrates the flow of program startup. The user will startup the program, which will create the Question Interface module. The user will be displayed with instructions about what the Strengths and Difficulties Questionnaire is and how to take it, as well as a button to begin the screener. Once that button is pressed, the Question Interface creates the Strengths and Difficulties Questionnaire module, which generates the survey and displays it.

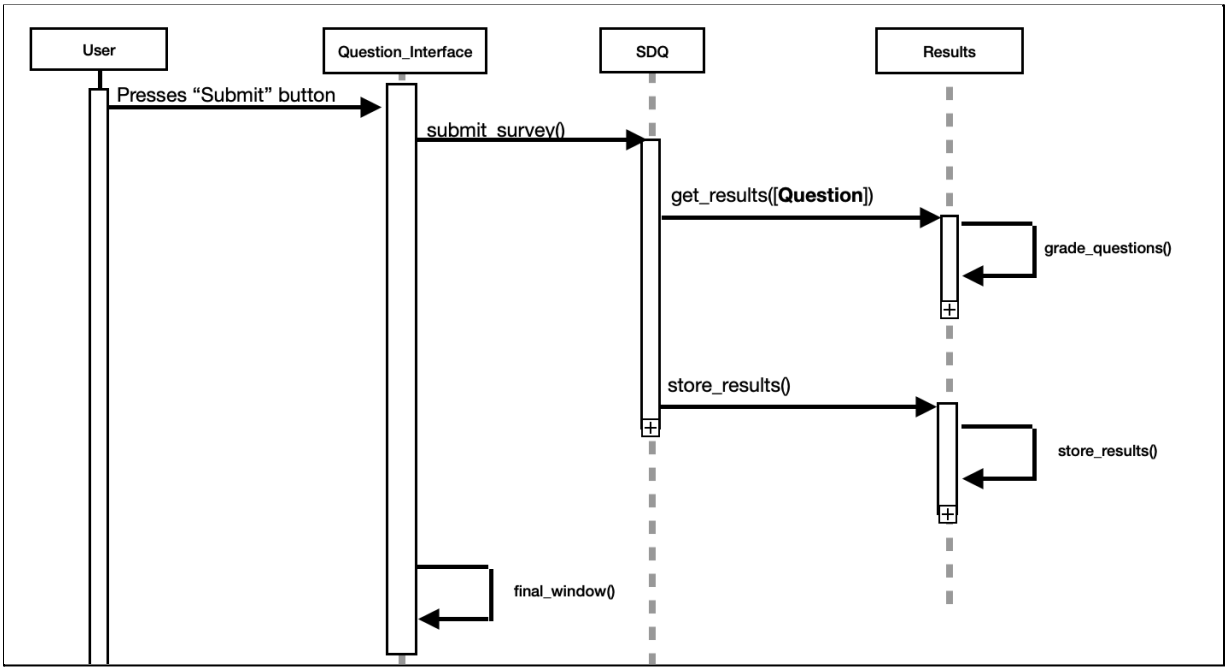


Figure 5.2: A UML Sequence Diagram of when a User Submits Their Answers

Figure 5.2 demonstrates the flow of answer submission. The user submits their survey, which prompts the Question Interface to send the submit() method to the Strengths and Difficulties Questionnaire. The Strengths and Difficulties Questionnaire calls methods on the results module to grade and store the questions. These questions are written to an output file.

6. References

IEEE Std 1016-2009. (2009). IEEE Standard for Information Technology—Systems Design—Software Design Descriptions. https://ieeexplore.ieee.org/document/5167255

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Commun. ACM, 15*(12), 1053-1058.

Hornof, Anthony. https://classes.cs.uoregon.edu/22W/cis422/Handouts/Template-SDS.pdf

7. Acknowledgements

This template is based off of Anthony Hornof's SDS template. This is cited in the references.