Design Specification Document Tutor Software

Ryan Czirr, Dylan Teare, Aaron Starr, Yiheng Lu

External Interface Descriptions & WND

Use Case 1: Log into Student Account

Inputs: Mouse and Keyboard

- The student navigates their mouse across the display and clicks inside the username form and password form and enters their username and password to be logged in.

Output: Display

- If the login is successful the student will be moved into a home window. If the login is unsuccessful the student will stay on the login window, and a small alert will state wrong “username or password”.

Use Case 2: Log into Tutor Account

Inputs: Mouse and Keyboard

- The tutor navigates the display using a mouse and uses the keyboard to enter text information to make a login attempt.

Outputs: Display

- Confirmation of a successful login is displayed to the tutor through access to the system. A failed login displays a failure message and asks the tutor for another input.

Use Case 3: Choose Subjects

Inputs: Mouse and Keyboard

- The student navigates the list of potential subjects and sub-subjects using their mouse and selects the desired choice by clicking. This process continues until the student finds their preferred sub-subject.
- The student enters their specific question using their keyboard as a means to accept text input. This allows the tutor to be aware of the student's exact need with the subject.

Outputs: Display

- The list of choosable subjects are displayed to the student monitor for ease of selection.

Use Case 4: Enter the Question

Inputs: Mouse and Keyboard

- The student will navigate into the enter the question form using their mouse, and then enter in a question into the form using their keyboard

Outputs: Display

- Once the student has entered their question they will navigate to a window where their question has been posted.

Use Case 5: Tutor browses open help requests

Inputs: Mouse

- Once the tutor is signed in the tutor will view a dashboard of open help requests. Once the tutor has found a help request they want to help with they can click on that help request.

Outputs: Display, Chat Window

- Once the tutor has selected and clicked on a help request they are moved from the help request dashboard to a new window with that help request opened, as well as a chat window with the student who asked for help

---

Use Case 6: Communicate

Inputs: Mouse and Keyboard

- Both users use the mouse the navigate the screen and interact with the display
- Both users use the keyboard to input messages to send to the other member of the chat

Outputs: Display, Chat Window

- Current and past messages between chat members are displayed on the chat window.
- All screen information is displayed on the user's monitor.

Use Case 7: End Chat

Inputs: Mouse and Keyboard

- Both users use either the mouse or keyboard to navigate the display.
- During a chat with a tutor, an *End Chat* button is clearly visible next to the chat window.
- Both users may use accessibility elements to navigate using only keyboard.

Output: Display, Chat Window

- After ending the chat, the *End Chat* button greys out and is non-intractable.
- After ending the chat, the chat window remains open

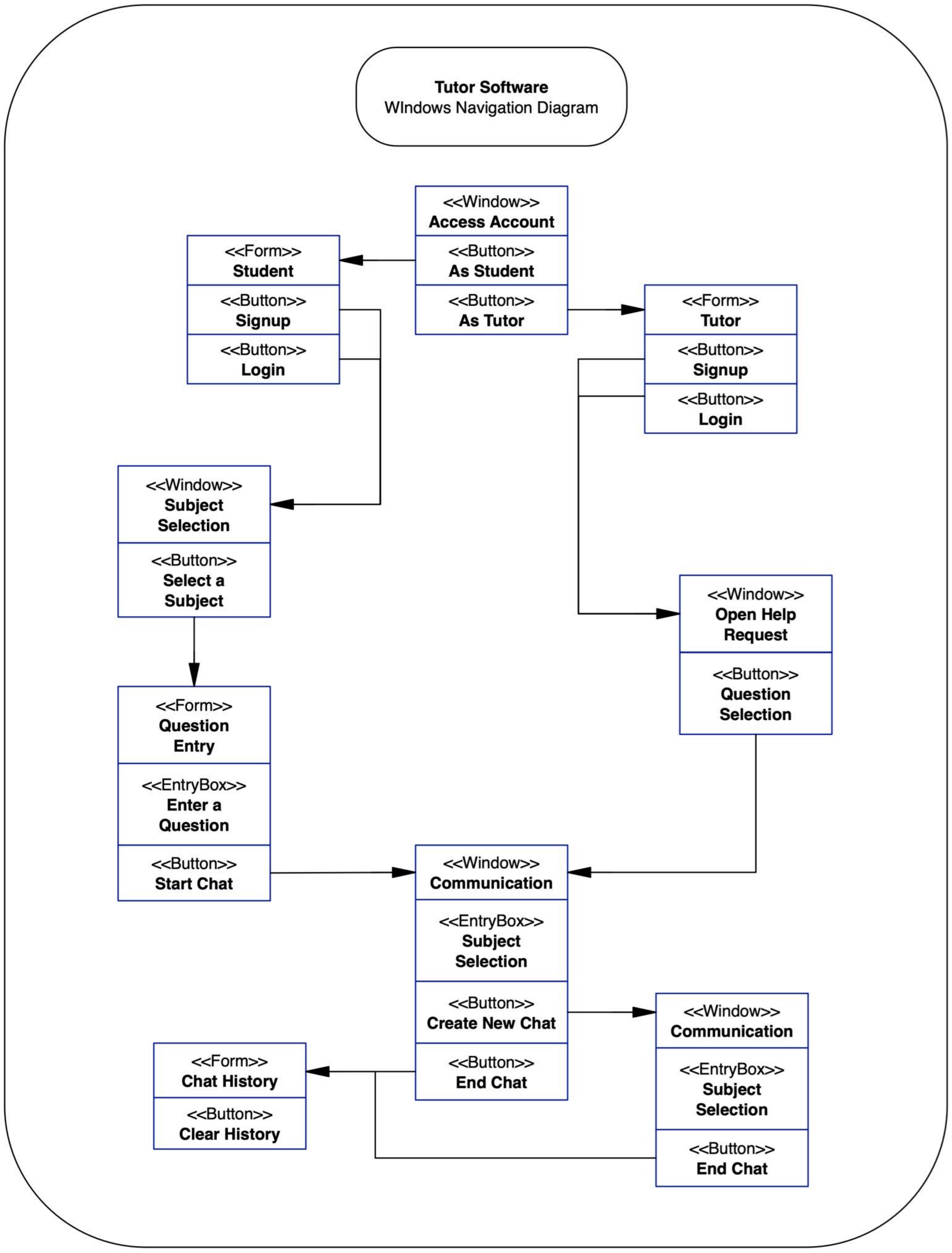
Use Case 8: Remove Chat History/Close Window

Inputs: Mouse and Keyboard

- Both users use either the mouse or keyboard to navigate the display.
- Both users can close the chat window with the X button in the top right of the window.
- Both users may use accessibility elements to navigate using only keyboard.

Output: Display, Chat Window

- Should the X be pressed after being disconnected, the chat window will close. The history of the chat is deleted.
- Should the X be pressed while still connected, the user will be prompted to confirm in the center of the screen. (focus this new window for accessibility). Confirming means the chat is disconnected, closed and deleted.


Tutor Software
WIndows Navigation Diagram
<<Window>> Access Account
<<Button>> As Student
<<Button>> As Tutor
<<Form>> Student
<<Button>> Signup
<<Button>> Login
<<Form>> Tutor
<<Button>> Signup
<<Button>> Login
<<Window>> Subject Selection
<<Button>> Select a Subject
<<Form>> Question Entry
<<EntryBox>> Enter a Question
<<Button>> Start Chat
<<Window>> Open Help Request
<<Button>> Question Selection
<<Window>> Communication
<<EntryBox>> Subject Selection
<<Button>> Create New Chat
<<Button>> End Chat
<<Window>> Communication
<<EntryBox>> Subject Selection
<<Button>> End Chat
<<Form>> Chat History
<<Button>> Clear History

Class Descriptions and Their Interactions

Class Name: Student Log In

Methods Called in Class:

- takeUserID(id): String

  - Description: Takes the user Id and matches it to a password taken from usePasswordEntry. If they match user is logged in

- usePasswordEntry(password): String

  - Description: Takes a password and matches it to a user id inputted from takeUserID and if they match the user is logged in.

Related Classes:

- Subject Selection
- Tutor Log In
- Chat Communication

Data to Other Classes:

- UserID: String

Data from Other Classes:

- N/A

Sequence of Interaction:

1. UserID + Password provided by user
2. Passed to Subject Selection class

Parent & Child Classes:

- N/A

Class Name: Tutor Log In

Methods Called in Class:

- takeUserID(id) : String

  - Description: Takes the user Id and matches it to a password taken from usePasswordEntry. If they match user is logged in

- userPasswordEntry(password) : String
    - Description: Takes a password and matches it to a user id inputted from takeUserID and if they match the user is logged in.

Related Classes:
- Student Log In
- Chat Communication

Data to Other Classes:
- UserID : String

Data from Other Classes:
- N/A

Sequence of Interaction:
1. UserID + Password provided by tutor
2. Passed to Chat Communication class

Parent & Child Classes:
- N/A

Class Name: Subject Selection

Methods Called in Class:
- broadSubjectSelection(subject) : String
    - Description: Allows the selection of a broad subject, narrows down user choice
- subTopicSelection(topic) : String
    - Description: Allows the selection of a subtopic within a subject
- questionInput(question) : String
    - Description: Collects the question input from a user to send to a tutor

Related Classes:
- Student Log In
- Chat Communication

Data to Other Classes:
- Subject : String
- SubTopic : String
- StudentQuestion : String

- Time : double

Data from Other Classes:
- Student Login - UserID : String

Sequence of Interaction:
1. UserID + Password required to access class
2. Send generated subject data to Chat Communication

Parent & Child Classes:
- N/A

Class Name: Chat Communication

Methods Called in Class:
- ChatWindowCreation(): void
    - Description: creates chat window.
- messagingCapabilities() :void
    - Description: allows users to message each other.
- endChat(): void
    - Description: ends chat between two users and closes the chat window.
- removeChatHistory(): void
    - Description: delete the chat history between two users.

Related Classes:
- Subject Selection
- Tutor Log In
- Student Log In

Data to Other Classes:
- N/A

Data from Other Classes:
- Student: Student Object
- Tutor: Tutor Object
- Subject: String
- Sub-Topic: String
- StudentQuestion: String
- Time: Double

Sequence of Interaction:

- For student users:
    - Log into student account
    - Select a subject, select a subtopic, enter a question to ask
    - Start a chat with a tutor on that question
- For tutor users:
    - Log into tutor account
    - Select a question
    - Start a chat with a student on that question

Parent & Child Classes:

- Parent Class: Subject Selection, Tutor Log In
- Child Class: N/A

Student Log in Methods:

Method Name: takeUserID	**Class Name:** Student Log In	**ID:** 1
Contract ID: 1	**Programmer:** John Doe	**Due Date:** 12/4/2021
Programming Language(s): Java		
Triggers/Events: Student wishes to login to the system, Student Log In class needs to read in the user ID.		
Arguments:	**Notes:**	
id : String	id variable contains a unique user ID.	
Sent & Received:	**Data Type:**	**Notes:**
Sent - N/A		
Receive - User ID	String	User ID information for login verification.
Arguments Returned:	**Notes:**	
N/A		
Algorithm Specification:		
See algorithm description		
Associated Use Cases: Log into Student Account		
Preconditions: None		
Postconditions: Student userID is submitted for login.		
Method Description: Gathers userID for student for later use in the system for verification.		

Algorithm Description:

takeUserID(id:String)

```
this.userID = id
IF this.attemptLogin(userID) == (failure)
    (collect new userID)
```

Method Name: userPasswordEntry	**Class Name:** Student Log In	**ID:** 2
Contract ID: 2	**Programmer:** John Doe	**Due Date:** 12/4/2021
Programming Language(s): Java		
Triggers/Events: Student wishes to login to the system, Student Log In requires user's password.		
Arguments:	**Notes:**	
password : String	password String contains the user's password for login.	
Sent & Received:	**Data Type:**	**Notes:**
Sent - N/A		
Receive - User Password	String	User password required for login verification.
Arguments Returned:	**Notes:**	
N/A		
Algorithm Specification:		
See algorithm description		
Associated Use Cases: Log into Student Account		
Preconditions: None		
Postconditions: Student password is submitted for login		
Method Description: Gathers password for student for later use in the system for verification.		

Algorithm Description:

```
userPasswordEntry(password)

this.userPassword = password
IF this.attemptLogin(userPassword) == (failure)
    (collect new user password)
```

Tutor Log in Methods:

Method Name: takeUserID	**Class Name:** Tutor Log In	**ID:** 3
Contract ID: 3	**Programmer:** John Doe	**Due Date:** 12/4/2021
Programming Language(s): Java		
Triggers/Events: Tutor wishes to login to the system, the tutor’s user ID is required for login verification.		
Arguments:	**Notes:**	
id : String	id String contains tutor’s unique user ID.	
Sent & Received:	**Data Type:**	**Notes:**
Sent - N/A		
Receive - id	String	User ID information for verification.
Arguments Returned:	**Notes:**	
N/A		
Algorithm Specification:		
See algorithm description		
Associated Use Cases: Log into Tutor Account		
Preconditions: None		

Postconditions: Tutor userID is submitted for login
Method Description: Gathers userID for tutors for later use in the system for verification.

Algorithm Description:

takeUserID(id):

```
this.tutorID = id
IF this.attemptLogin(tutorID) == (failure)
    (collect new tutor ID)
```

Method Name: userPasswordEntry	**Class Name:** Tutor Log In	**ID:** 4
Contract ID: 4	**Programmer:** John Doe	**Due Date:** 12/4/2021
Programming Language(s): Java		
Triggers/Events: Tutor wishes to login to the system, the tutor's password is required for login verification.		
Arguments:	**Notes:**	
password : String	password contains the tutor's password associated with their account.	
Sent & Received:	**Data Type:**	**Notes:**
Sent - N/A		
Receive - password	String	Tutor password for login verification.
Arguments Returned:	**Notes:**	
N/A		
Algorithm Specification:		

See algorithm description
Associated Use Cases: Log into Tutor Account
Preconditions: None
Postconditions: Tutor password is submitted for login
Method Description: Gathers password for tutor for later use in the system for verification

Algorithm Description:

userPasswordEntry(password)

```
this.tutorPassword = password
IF this.attemptLogin(tutorPassword) == (failure)
    (collect new tutor password)
```

Subject Selection Methods:

Method Name: broadSubjectSelection	**Class Name:** Subject Selection	**ID:** 1
Contract ID: 5	**Programmer:** John Doe	**Due Date:** 12/4/2021
Programming Language(s): Java		
Triggers/Events: Student is looking to ask a question. Selects the dropdown for broadSubjectSelection		
Arguments:	**Notes:**	
Subject : string	The name of the broad subject.	
Sent & Received:	**Data Type:**	**Notes:**
Sent - Database Query	string	Sends a query to the database so the subTopicSelection dropdown has a populated list of options

Received - Database	list	subTopicSelection options based on broadSubjectSelection
Arguments Returned:	**Notes:**	
Success - int	Returned whether the query was successful or not	
Algorithm Specification:		
See algorithm description		
Associated Use Cases: 2		
Preconditions: Student is logged in, Dropdown option is selected for broadSubjectSelection		
Postconditions: Database returned list for subTopicSelection		
Method Description: System displays a dropdown menu to allow for students to make a selection of available question options and gathers the input		

Algorithm Description:

```
returnedVal = -1
If button.dropdown == selected:
    -   Display all broadSubjectSelection options
    -   dataReturn = sendDatabaseQuery(broadSubjectSelection)
    -   If dataReturn:
            -   returnedVal = 0 (success)
    -   Else:
            -   returnedVal = 1 (failure)
Return returnedVal
```

Method Name: subTopicSelection	**Class Name:** Subject Selection	**ID:** 2
Contract ID: 6	**Programmer:** John Doe	**Due Date:** 12/4/2021
Programming Language(s): Java		
Triggers/Events: Student is looking to ask a question. Selects the dropdown for subTopicSelection after broadSubjectSelection.		
Arguments:	**Notes:**	

Subject : string	The name of the sub topic subject.	
Sent & Received:	**Data Type:**	**Notes:**
Received - topic	string	The subTopicSelection from the broadSubjectSelection returned list
Arguments Returned:	**Notes:**	
Success - int	Returned whether the selection was successful or not	
Algorithm Specification:		
See algorithm description		
Associated Use Cases: 2		
Preconditions: Student is logged in, Dropdown option is selected for broadSubjectSelection, Database returned list for subTopicSelection, Dropdown option is selected for subTopicSelection		
Postconditions: Question input field is visible		
Method Description: System displays a dropdown menu for sub-topic selection by the student and gathers the input		

Algorithm Description:

```
returnedVal = -1
If button.dropdown == selected && broadSubjectSelection.returned == True:
    -  Display broadSubjectSelection.list
    -  returnedVal = 0 (success)
Else:
    -  returnedVal = 1 (failure)
Return returnedVal
```

Method Name: questionInput	**Class Name:** Subject Selection	**ID:** 3
Contract ID: 7	**Programmer:** John Doe	**Due Date:** 12/4/2021
Programming Language(s): Java		
Triggers/Events: Student is looking to ask a question. Types question after subTopicSelection and broadSubjectSelection.		

Arguments:	Notes:	
Question : string	The question to be asked to a tutor	
Sent & Received:	**Data Type:**	**Notes:**
Received - question	string	The student's written question
Arguments Returned:	**Notes:**	
Success - int	Returned whether the question input was successful or not	
Algorithm Specification:		
See algorithm description		
Associated Use Cases: 2		
Preconditions: Student is logged in, Dropdown option is selected for broadSubjectSelection, Database returned list for subTopicSelection, Dropdown option is selected for subTopicSelection, Question input field is visible		
Postconditions: Question submitted for tutor review		
Method Description: questionInput() allows students to input questions		

Algorithm Description:

```
returnedVal = -1
If subTopicSelection.returned == True && broadSubjectSelection.returned == True:
    -   Display input field for user question
    -   User presses submit
    -   returnedVal = 0 (success)
Else:
    -   returnedVal = 1 (failure)
Return returnedVal
```

Chat Communication Methods:

Method Name: chatWindowCreation()	**Class Name:** Chat Communication	**ID:** 1
Contract ID: 8	**Programmer:** John Doe	**Due Date:** 12/4/2021
Programming Language(s): Java		

Triggers/Events: Student or tutor wishes to chat with another student or tutor		
Arguments:	**Notes:**	
N/A	No argument is passed to this method.	
Sent & Received:	**Data Type:**	**Notes:**
N/A	N/A	No argument sent/received
N/A	N/A	No argument sent/received
Arguments Returned:	**Notes:**	
N/A	No argument is returned. The method has a void return type.	
Algorithm Specification:		
See algorithm description		
Associated Use Cases: communicate		
Preconditions: Student or Tutor clicks the chat button		
Postconditions: A chat window is created		
Method Description:		
chatWindowCreation() creates a chat window that allow student and tutor to communicate		

Algorithm Description:

chatWindowCreation():

```
If (chat button is clicked){
    Create chat window;
}
```

Method Name: messagingCapabilities()	**Class Name:** Chat Communication	**ID:** 2
Contract ID: 9	**Programmer:** John Doe	**Due Date:** 12/14/2021
Programming Language(s): Java		
Triggers/Events: When the tutor or student wants to message another tutor or student using the chat window that has been created previously.		
Arguments:	**Notes:**	
N/A	No argument is passed to this method.	
Sent & Received:	**Data Type:**	**Notes:**
Sent - N/A	N/A	No argument sent/received
Received - N/A	N/A	No argument sent/received
Arguments Returned:	**Notes:**	
N/A	No argument is returned. The method has a void return type.	
Algorithm Specification:		
See algorithm description		
Associated Use Cases: Communicate		
Preconditions: Student or tutor types in a message and sends it to the chat window with return keystroke		
Postconditions: Their message is sent and displayed in the chat window		
Method Description:		
messagingCapabilities() allows the students and tutors to send messages to each other		

Algorithm Description:

messagingCapabilities():

```
chatWindow = chatWindowCreation();

text = keystrokes from keyboard till return is hit;

chatWindow.message(text);
```

Method Name: endChat()	**Class Name:** Chat Communication	**ID:** 3
Contract ID: 10	**Programmer:** John Doe	**Due Date:** 12/4/2021
Programming Language(s): Java		
Triggers/Events: User wants to terminate the chat and exit the chat room		
Arguments:	**Notes:**	
N/A	No argument is passed into the method	
Sent & Received:	**Data Type:**	**Notes:**
N/A	N/A	No argument sent/received
N/A	N/A	No argument sent/received
Arguments Returned:	**Notes:**	
void	The method has a void return type	
Algorithm Specification:		
See algorithm description		
Associated Use Cases:		
chatWindowCreation() //the method needs to have a window to close		
Pre-condition:		

There is a chat online; User clicks end chat
Post-Condition:
Chat room closed; Notify message emerge on screen; Update chat history
Method Description:
endChat() terminates a chat and close the chat window

Algorithm Description:

If button endChat is clicked: - endChat() //terminate the chat - Close chat room - Display “Chat ended” to screen - Update chat history

Method Name: removeChatHistory()	**Class Name:** Chat Communication	**ID:** 4
Contract ID: 11	**Programmer:** John Doe	**Due Date:** 12/4/2021
Programming Language(s): Java		
Triggers/Events: User wish to clean up the chat history		
Arguments: N/A	**Notes:** No argument is passed into the method	
Sent & Received:	**Data Type:**	**Notes:**
N/A	N/A	No argument sent/received
N/A	N/A	No argument sent/received
Arguments Returned:	**Notes:**	
void	The method has a void return type	

Algorithm Specification:
See algorithm description
Associated Use Cases:
N/A //No associated use cases
Pre-conditions:
User click clear history button
Post-conditions:
Clear chat history; Display message into screen
Method Description:
removeChatHistory() clears all the previous historical data of chats

Algorithm Description:

```
If button clear history is clicked:
    -   removeChatHistroy() //Clear current chat history
            -   If fail to clear history: display “unable to clear history” into screen
            -   If history cleaned: display “chat history cleared” into screen
```

Database Information:

Database Software: MongoDB
Purpose: Store Student account data, Store Tutor account data, Store Student questions, Store Tutor answers
Organization Method: Store data in different tables based on Purpose. Correlate data based on unique identifier for each user (Student or Tutor)
Information Columns: (primary keys underlined)

- Store Student account data:
    - <u>StudentUniqueID</u>, StudentUsername, StudentPassword (hashed)
- Store Tutor account data:
    - <u>TutorUniqueID</u>, TutorUsername, TutorPassword (hashed)
- Store Student questions:
    - <u>StudentUniqueID</u>, StudentUsername, StudentQuestion, TutorUniqueID
- Store Tutor answers:
    - <u>TutorUniqueID</u>, TutorUsername, TutorReponse, StudentUniqueID