Software Design Specification

The Hub

Revision 1.0

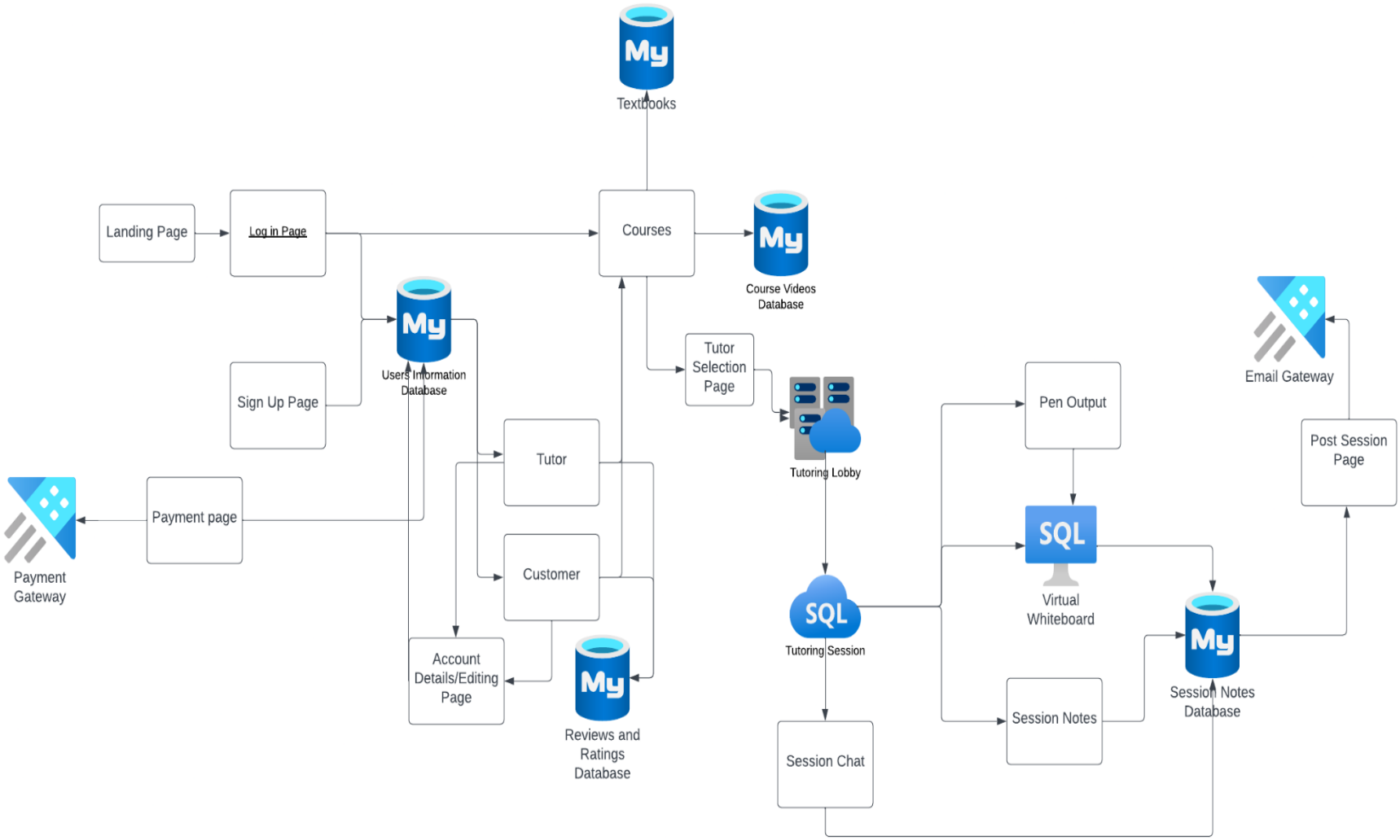
Raymond Abayon | Steven Bernas | Andrew Wu

System Description

System Overview

- The Hub is different from the rest of the online tutoring application services is th
- at our software application has real time video chat tutoring and includes a bluetooth drawing pad that connects to PC, Android and MacOs devices with a monthly or annual subscription
- The Hub is launching the software application to college students with an .edu email first, student can request an instant live tutor 24 hours a day with a wait time between 5- 10 minutes since we have tutors in different time zones, students K-12 will be launched at a later date
- The Hub uses an external bluetooth pencil and drawing pad that is compatible with Android, iOS, Windows and MacOS that is included with the monthly or annual subscribers
- The Hub's interface uses split screens that can zoom in and out of the white board or video cam of the user's preference
- The Hub's interface dashboard show's user's school subject icons, subscription upgrade icon, subject forum's icon, dashboard settings icon for page preference and profile settings icon
- After each tutoring session, a transcript of the meeting and recorded video will be emailed or texted to students along with the whiteboards and notes taken during the tutoring session.
- Application built with C++ and server is build on MySQL

Architectural Diagram



Architectural Diagram Description

Architectural Overview

- This software architecture diagram is the high level view of the software we will develop for The Hub, our online tutoring service.
- The components contain many different interfaces to have the customer interact with the service.
- The interfaces will be different web pages taking the form of the white boxes within the diagram.
- There are five different databases depicted that will hold user information, textbooks, reviews, notes, and course videos.
- Each database will be accessed by the system through the varying interfaces throughout the software.
- One important matter is that there will be a layer of security for connection to the user database since it contains sensitive information.
- The data within that database will be encrypted as well.
- The diagram also contains two gateways such as the payment gateway to connect the software with payment methods and an email gateway to send the notes to customer's email accounts.
- There will also be a multimedia router to connect everyone together and connect other functional services such as the whiteboard and the actual session window.
- Creating a detailed software architecture diagram directly within this text-based interface is quite complex and limited.
- However, I can describe the typical elements and structure you might include in a software architecture diagram.
- A software architecture diagram typically represents the high-level structure of a software system, illustrating the various components, their interactions, and the overall layout. Here are some common elements you might include below.

Components

- Represent different parts of the system, such as user interface (UI), application logic, database, external services, etc.

Connections/Interfaces

- Illustrate how components communicate with each other or with external systems.
- This could include APIs, message queues, direct method calls, etc.

Layers

- Show the separation of concerns, such as presentation layer (UI), business logic layer, and data storage layer.

Deployment Diagram

- Illustrate how the software is deployed across various servers or hardware, including cloud services, on-premises servers, etc.

Frameworks/Libraries

- Indicate any third-party frameworks or libraries used within the system.

Databases

- Represent the databases used, their types (relational, NoSQL), and how they interact with the application.

Security

- Show security components like authentication, authorization mechanisms, and how sensitive data is handled.

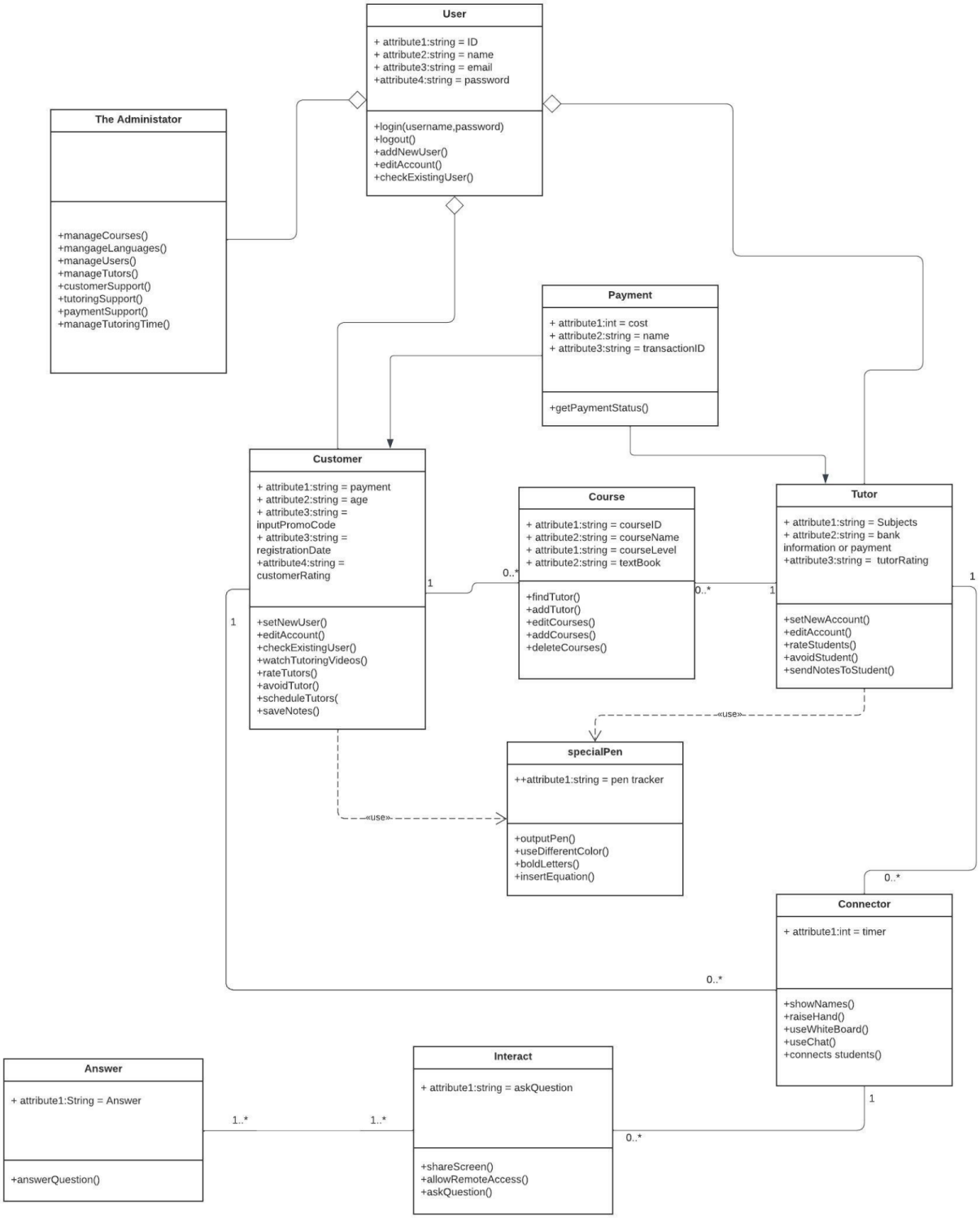
External Systems

- Depict other systems or services that interact with the software, whether they are APIs, external databases, or other applications.

Annotations and Notes

- Provide additional information and explanations for the components and connections.
- It's important to tailor the diagram to your specific software design and architecture.
- Use appropriate diagramming tools such as draw.io, Lucidchart, or UML tools like Enterprise Architect or Visual Paradigm to create a clear and visually appealing software architecture diagram based on the elements mentioned above.

UML Class Diagram



Class Overview

User Class Attributes

- Name (string)
- Identification (ID) (string)
- Email Address (string)
- Password (string)
- Methods:
- Login(username: string, password: string)
- Logout()
- EditAccount()
- CheckExistingUser()

Tutor Class (Inherits from User) Attributes

- Subjects Specialized In (string)
- Payment Information (string)
- Hidden Rating Score (string)
  Methods:
- SetNewAccount()
- EditAccount()
- RateTutors()
- SendNotesToStudents()

Customer Class (Inherits from User) Attributes

- Payment Method (string)
- Age (string)
- Date of Registration (string)
- Special Promo Codes (string)
- Hidden Score (string)
  Methods:
- CreateNewAccount()
- EditAccount()
- CheckCustomerIDUsed() (bool)
- RateTutor()
- AvoidTutor()
- ScheduleAndRequestTutors()

- SaveNotesAndSendByEmail()

Administrator Class (Inherits from User) Methods

- ManageCourses()
- ManageLanguage()
- ManageUsers()
- TutoringSupport()
- CustomerSupport()
- PaymentSupport()
- ManageTutoringTime()

Payment Class Attributes

- Cost of Tutoring Session (int)
- Name of Parties (string)
- Transaction Identification (string)
- Methods:
- GetPaymentStatus()

Course Class Attributes

- Course ID (string)
- Course Name (string)
- Course Level Difficulty (string)
- Required Textbook (string)
  Methods
- FindTutor()
- AddTutor()
- EditCourse()
- AddCourse()
- DeleteCourse()

SpecialPen Class Attributes

- Pen Tracker (string)
- Methods:
- OutputPen()
- UseDifferentColor()
- BoldLetters()
- InsertEquation()

Connector Class Attributes

- Countdown of Duration (int)
- Methods:
- ShowNames()
- RaiseHand()
- WhiteBoard()
- UseChat()
- ConnectStudents()

Interact Class Attributes

- Ask Questions (string)
- Methods:
- ShareScreen()
- RemoteAccess()

Answer Class Attributes

- Answer (string)
- Methods:
- AnswerQuestion()

Attributes Description

User Class

- String attribute 1: This attribute is of the String data type and will be the ID or username of the customer.
- String attribute 2: This attribute will be of the String data type and will be the user's legal name.
- String attribute 3: This attribute will be a String data type and will collect the user's email address for use with their account.
- String attribute 4: This attribute will be of a String data type and will contain the user's password.

Customer Class

- String attribute 1: This attribute of string data type will contain the user's payment method.
- Int attribute 2: This attribute will be of an int data type and will contain the user customer's age.
- String attribute 3: This attribute will be a string data type and will contain a promo code the customer can enter for special deals.
- String attribute 4: This attribute will be a string that will have the registration date of the user.

Payment Class

- Attribute 1: This attribute will have the cost of the payment that is due for the user.
- String attribute 2: This attribute will have the name of the account holder to whom the payment is due for.
- String attribute 3: This attribute will contain an ID that will be used to reference the specific transaction.

Course Class

- String Attribute 1: This attribute will contain the course ID in the form of a string.
- String attribute 2: This attribute will contain the course's name in the form of a string.
- String attribute 3: This attribute will contain the difficulty level in a categorical form, data type being a string.
- String attribute 4: Will contain the textbook name for the corresponding course.

Tutor Class

- String attribute 1: This attribute will contain the subjects that the specific tutor specializes in.
- String attribute 2: This attribute will contain the bank information/ money transfer information that is corresponding with the tutor account.
- String attribute 3: This attribute will contain the rating that is given to the tutor.

Special Pen Class

- Attribute 1: This attribute will contain the tracker for the pen. Will be attributed with the input device the user is using. Ex. Mouse, trackpad, tablet, etc.

Connector Class:

- Int attribute 1: This attribute will contain the running minutes of the session.

Interact Class:

- String attribute 1: This string will contain the question that the customer will be asking the tutor.

Answer Class:

- String attribute 1: This string will contain the answer for the specific question that was asked by the customer.

Operations Description

Administrator Class

- manageCourses: This function will allow the administrator to update and change courses within the software.
- manageLanguages: This function will allow the administrator to add and remove string inputs that will categorize different languages that the session will be using.
- manageUsers: This function will allow the administrator to edit user profiles along with managing and deleting them.
- manageTutors: This function will allow the administrator to edit tutor profiles along with deleting them.
- customerSupport: This will allow the administrator to look into customer support questions with questions that they may have inputted. This function may prompt a chat box between the user and the administrator.
- tutoringSupport: This will allow the administrator to look into issues tutors may be having and assist. This function may prompt a chat box to troubleshoot issues that the tutor may be having. Will intake string inputs along with chars for communication.
- paymentSupport: This will allow the administrator to contact 3rd party payment actors to troubleshoot issues they may have. This function will intake strings and chars as a means of communication between both parties.
- manageTutoringTime: This function will open a chart with the different times that tutors will have to host their sessions. Administrators will have the ability to delete values and other data types that the tutor user may input into the chart. Administrators will also have the ability to add to this chart as well, taking in integers to fill in attributes for the times.

User Class

- Login: This function will allow the user to log in. It will intake 2 strings, one for the password and one for the username. This will come in the form of a sign in page. Upon checking the database if the inputs are equivalent, the user will be signed in. Otherwise it will prompt the user to enter information again.
- Logout: This function will allow the user to log out of their account from the software. This will come in the form of a button along the top or bottom edge of the page.
- addnewUser: This function will intake 4 strings that will account for the user's legal name, userID, password, and email address. The interface will be in the form of a new sign up page that will be connected to the login and landing page. This will then add the information to the user database for storage and encrypted to protect user information.

- editAccount: This function will allow the user to edit attributes that are connected to their account. This will take form as a new page where users can see all the attributes inputted and change them as needed.
- checkExistingUser: This function will check if the user created account matches with others in the database previously. It will compare strings of the user by looking at the user's name and userID to see if the user is already in the system.

Customer Class

- setNewUser: This function will set the user account as a member of the customer class. Will most likely take form as a selection bubble on the signup page.
- editAccount: This function will invoke the editAccount function from the user class.
- checkExistingUser: This function will invoke the checkExistingUser from the user class.
- watchTutoringVideos: This function will come in the form of a selection on the landing page that will allow the customer to traverse the database/library of tutor made videos on different courses and subjects.
- rateTutors: This function will allow the customer to rate a specific tutor that they are connected with. This will intake a string that will categorize the first character as a number rating followed by a comment that will contribute to the tutor. This will take the form of a button selection that will bring up an input box and prompt the user on how to format their rating correctly.
- avoidTutors: This function will allow the customer to add the tutor to a form of a block list so the customer and tutor will not be paired up again in the future. This will simply add the tutors name to the customers block list and if the strings match up on a condition check, they will not be paired. The interface will be in the form of a block button in the tutoring session.
- scheduleTutors: This function will prompt the user to schedule a time to have a session with a specific tutor. This will come in the form of a new page where the customer can browse and filter between courses and availability time, or even sort through a list of tutors. This will intake strings for the filters.
- saveNotes: Customers will be able to save notes that were provided by the tutor during the tutoring session. This will output a pdf and send it to the customer's email linked to their account. This will come in the form of a button press during the tutoring session window.

Tutor Class

- setNewAccount:This function will set the user account as a member of the tutor class. Will most likely take form as a selection bubble on the signup page. No further input needed.
- editAccount: This function will invoke the editAccount function in the user class.
- rateStudents: This function will allow the tutor to rate students based on experience. This will take the form of an input window after a button press to prompt the rate system. This will collect a string that will categorize the first character as a rating and the rest to be a comment following the rating. This string input will be stored in the reviews database connected with the customer.
- avoidStudents: This function will allow the tutor to add students to their own block list in case the customer provided an unpleasant experience. This will add the customer to the tutors list (potentially string array) and if they match in the future, the session will not be allowed to connect and will prompt the reason.
- sendNotesToStudent: This function will save the notes in the tutoring sessions window as a pdf and send it to the customer's email address in the system. This will come in the form of a button press in the tutoring window or as a prompt after the session has ended.

Payment Class

- getPaymentStatus: This function will allow customers to check the status of their payments. Most likely there will be a drop down selection with the payment ID upon the payment page and will only be able to view the status provided by the 3rd party payment processor and no edits will be allowed.

Course Class

- findTutor: This function will find tutors that will be able to tutor a certain course. This will intake a string and compare with the list of tutors within the system. From here the customer will be able to schedule an appointment.
- addTutor: This function will add tutor accounts to the list of applicable tutors able to assist in the course. This will intake a string and add it to the list (array) of applicable tutors. This function will most likely be accessible to the administrator and tutor classes and will allow them to edit the list. The function will take the form of a input box viewable to the administrator and tutor classes to add the tutors to the list.
- editCourses: This function will allow administrators to edit the course details and attributes of the courses. This will take the form of a new page where the courses will be displayed and the attributes can be edited via clicking on the data and changing them.
- addCourses: This function will allow the administrator to add courses to the system. This will intake 4 strings to provide a new course ID, name, and level,

along with the textbook name suitable for the course. This will take form as a drop down menu in the courses page accessible by the administrator.
    - deleteCourses: This function will allow the administrator to delete courses from the system. This will take the form of a delete button next to the individual courses displayed in the courses page. The administrator will be the only account type with this privilege.

Special Pen Class

- outputPen: This function will work with the input device to provide a way for the pen to be visible on the virtual whiteboard. It will create a line whenever the input device is dragging and clicking inputs are being read.
- useDifferentColor: This function will change the color of the output coming from the pen. Different colors will be selected via a drop down menu and the tutor or customer will be able to select which they would like to use.
- boldLetters: This function will embolden the output of the pen. The user will be able to highlight parts of the virtual board and make the trace path thicker for the trails that are within the highlighted area. This function will be activated with a button press on the tutoring session window.
- insertEquations: This function will allow users to type and input mathematical equations on the note sheet or whiteboard. This will intake char and string inputs from the users and will be activated by a button press in the tutoring session window.

Connector Class

- showNames: This function will display the names of the users connected to the tutoring session. This will not take an input but will display the names of all users via a pop up window when pressed by the button in the tutoring session window.
- raiseHand: This function will display a hand connected with the customers name to the tutor to indicate they are raising their hand. This will be activated by a button press on the tutoring window through the customer side.
- useWhiteBoard: This function will launch the virtual whiteboard. This function will expand the window of the tutoring session so users can draw, write or type on this interface. The function will be activated by the user in the form of a button along the tutoring session window.
- useChat: This function will launch a separate mini window displaying 2 boxes. One for the chat output and one for the user input. This will intake strings and chars from the user and the message will be displayed along the output window with their name by the message. This function will be activated via button press within the tutoring session window.
- connectsStudents: This function will allow the tutor to grant access to students in the waiting area to the tutoring session. This will come in the form of a window

before the tutoring session and will have buttons accompanying the students name to deny or accept them into the session.

Interact Class

- sharesScreen: This will allow the customer or tutor to share their computer screen. This will display the user's computer screen and will be activated by a button on the tutoring window. This will prompt the host and the host will allow access to the one who activated the function.
- allowRemoteAccess: This function will allow the customer to give computer access to the tutor. This will only be accessible when shareScreen is active. This will be in the form of a button press along the tutoring session window.
- askQuestions: This function will allow the customer to ask the tutor a question. This function will prompt an input box for the customer to send to the tutor. This message will be sent to the tutor to invoke the answerQuestion function. This will be activated by a button press in the chat window. This will intake strings from the customer.

Answer Class

- answerQuestion(): This function will allow the tutor to answer private questions from the customers. The question from the customer will be displayed and an input box will pop up to prompt the tutor to enter the answer to the question from the customer. This will intake strings from the user.

Development Plan

Overview Description

- The Hub software application's first priority is to launch to students at universities and community colleges that contain a .edu email
- Second priority is to launch to students K-12
- Estimation of deployment completion is in 2 months, to meet the 2 month deployment completion mark, some of our team members need to have daily sync up meetings with stakeholders in order to meet the specifications while other parts of team is working on software development in parallel to use our time efficiently to meet our deployment milestone date

Project Tasks

- Steven, Raymond, Andrew are responsible for making sure the Initial Planning, Design, Development and Deployment is completed by milestone date marked on the mark
- Each step needs to be implemented carefully and efficiently and reviewed by stakeholders for approval

Initial Planning

- Daily meetings with team to meet specific requirements by stakeholders
- Daily meetings with stakeholders to make sure standards are met
- Documenting system requirements and meeting minutes for reference
- Completing the goal by milestone date per Gantt chart attached

Design

- Designing the database to handle millions of users and need to compatible with 1GB/S to handle real time tutoring on the application or PC
- Trying to implement software to use Big O complexity of O(1) to make application browsing using the application or website is smooth and fast as possible
- Designing the interface to be simple, easy to use and suits ages
- Confirming stakeholders approve of database, software, interface demonstration and documented by Raymond for reference
- Completing the goal by milestone date per Gantt chart attached

Development

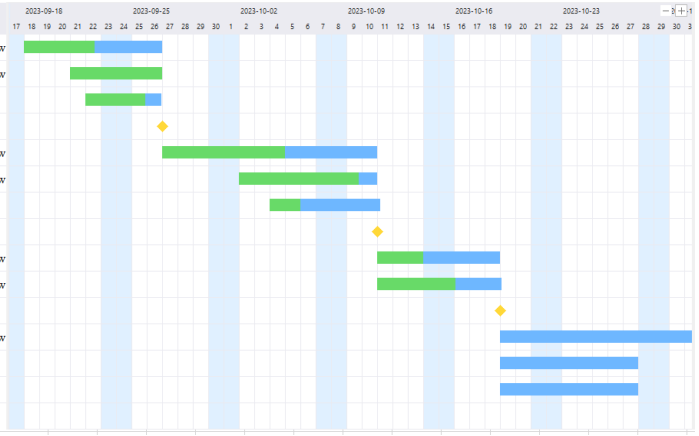
- After database, software, interface is implemented with The Hub application, our team can perform pretesting, which is checking the functionality
- Is the software system doing what it is supposed to be?
- Is the call between the mobile to mobile, mobile to PC or PC to mobile clear with any lag time?
- Does the pen write smoothly without any lag time and the connection fast enough to write in real time?
- Does the software system work with any browser on PC or mobile device?
- Does the software system work with Android, iOS, Windows and MacOS?
- Andrew will log all issues that arise and communicate with our team for mitigation
- After all these pretests pass we can move to deployment if any changes need to be made by stakeholders a ECO (Engineering Change Order) needs to be submitted
- Completing the goal by milestone date per Gantt chart attached

Deployment

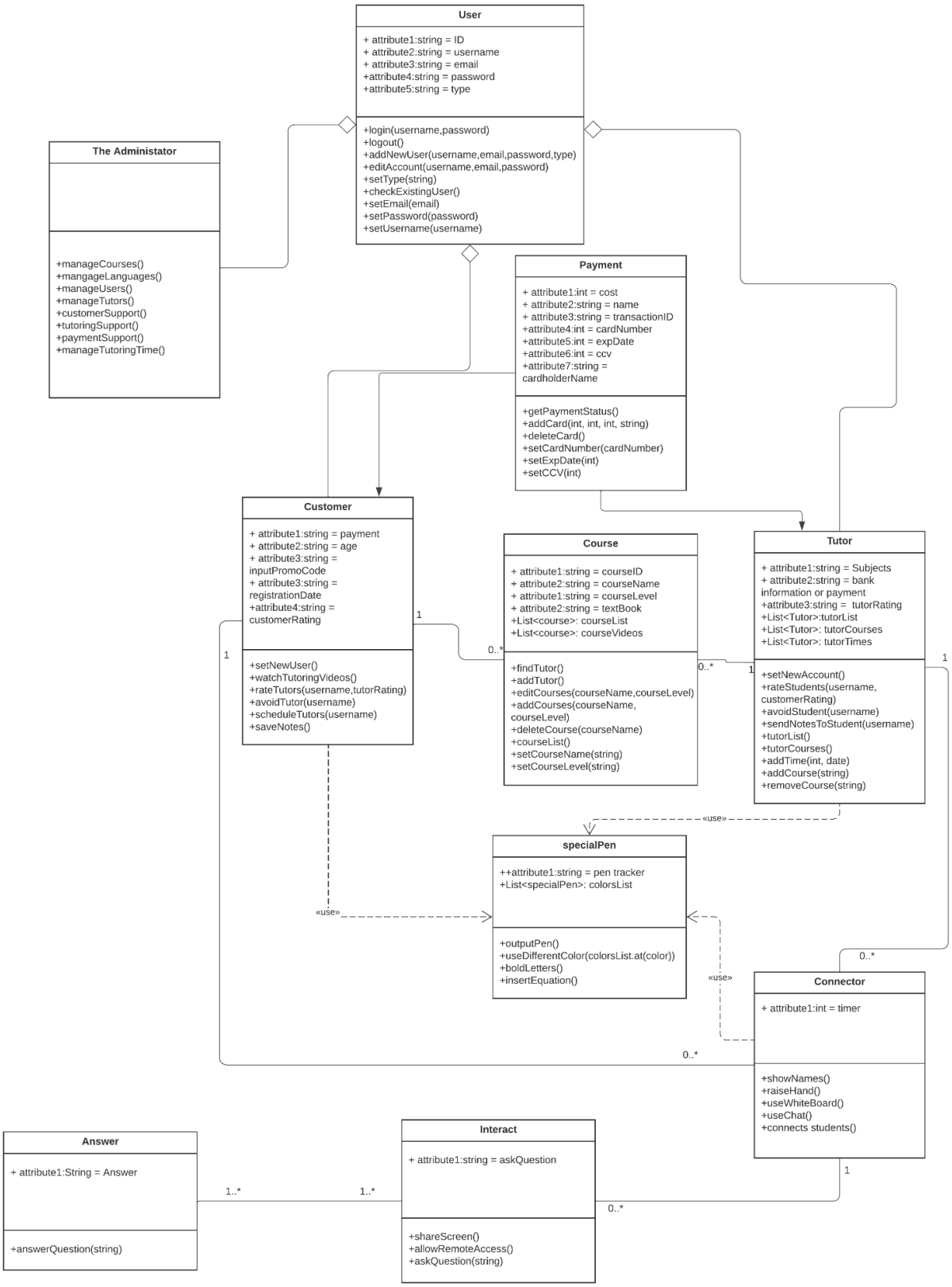
- After pretesting our team will perform posttesting to make sure The Hub application is robust, does not crash or break and maintains a reliable software application
- After hours of extensive testing, do we encounter any hardware malfunctioning with our software?
- Are unwanted inputs in the application still producing the correct output during post unit software testing
- If there are any issues or hardware software compatibility mismatch, do we have a work around to deploy the The Hub application while it is in the field?
- Raymond will monitor post testing and will report to our team for mitigation
- This is the critical milestone we need to meet in a 2 month time frame

Timeline

ID	Task	Start	Finish	Duration	Progress	Priority	Resources
1	Initial Planning	2023-09-18	2023-09-26	7 d	65%	1	Steven, Raymond, Andrew
2	Team/Stakeholders Meeting	2023-09-21	2023-09-26	4 d	75%	1	Steven, Raymond, Andrew
3	Document System	2023-09-22	2023-09-26	3 d	67.1%	3	Andrew
4	Initial Planning Completed	2023-09-18	2023-10-05	14 d	100%	4	
5	Design	2023-09-27	2023-10-10	10 d	60%	1	Steven, Raymond, Andrew
6	Database/Software/Interface	2023-10-02	2023-10-10	7 d	82%	1	Steven, Raymond, Andrew
7	Design Specifications	2023-10-04	2023-10-11	5.2 d	38%	4	Raymond
8	Design Completed	2023-09-27	2023-10-24	20 d	60%	2	
9	Development	2023-10-11	2023-10-18	6 d	50%	1	Steven, Raymond, Andrew
10	Integrate/Develop Systems/Pretest	2023-10-11	2023-10-19	6.1 d	50%	2	Steven, Raymond, Andrew
11	Development Completed	2023-10-11	2023-10-26	12 d	50%	3	Andrew
12	Deployment	2023-10-19	2023-11-01	10 d	0%	1	Steven, Raymond, Andrew
13	System Maintence/Evaluaton	2023-10-19	2023-10-27	7 d	0%	4	Andrew
14	System Postesting	2023-10-19	2023-10-27	7 d	0%	2	Raymond
15	Deployment Completed	2023-10-17	2023-11-17	24 d	0%	1	



Updated UML Diagram and Description



Class Descriptions (Updated)

Reasons for changes: The reason for the change in UML diagram is to add setter and getter methods to the classes so as to be able to set attributes properly and allow for editing. Another reason is to add more attributes to certain classes such as vector arrays that will contain information for the different classes. These changes were necessary in order for the software to function properly as well as testing to be done thoroughly.

User Class Attributes

- Name (string)
- Identification (ID) (string)
- Email Address (string)
- Password (string)
- Type(string)
- Methods:
- Login(username: string, password: string)
- Logout()
- addNewUser(username:string, email:string, password:string)
- EditAccount(username:string, email: string, password:string)
- setType(string)
- CheckExistingUser()
- setEmail(email:string)
- setPassword(string)
- setUsername(string)

Tutor Class (Inherits from User) Attributes

- Subjects Specialized In (string)
- Payment Information (string)
- Hidden Rating Score (string)
- Vector Tutor List <string>
- Vector Tutor courses <string>
- Vector Tutor times <datetime>

  Methods:
- SetNewAccount()
- RateStudent()
- avoidStudent(string)
- SendNotesToStudents()
- tutorList()
- tutorCourses()
- addTime(int,datetime)

- addCourses(string)
- removeCourse(string)

Customer Class (Inherits from User) Attributes

- Payment Method (string)
- Age (string)
- Date of Registration (string)
- Special Promo Codes (string)
- Hidden Score (string)

  Methods:
- CheckCustomerIDUsed() (bool)
- watchTutoringVideos()
- RateTutor(string, int)
- AvoidTutor()
- ScheduleTutors()
- SaveNotesAndSendByEmail()

Administrator Class (Inherits from User) Methods

- ManageCourses()
- ManageLanguage()
- ManageUsers()
- TutoringSupport()
- CustomerSupport()
- PaymentSupport()
- ManageTutoringTime()

Payment Class Attributes

- Cost of Tutoring Session (int)
- Name of Parties (string)
- Transaction Identification (string)
- Card Number (int)
- Expiration Date (int)
- CCV (int)
- Cardholder Name (string)
- Methods:
- GetPaymentStatus()
- addCard(int, int, int, string)
- deleteCard()
- setCardNumber(int)
- setExpDate(int)

- setCCV(int)

Course Class Attributes

- Course ID (string)
- Course Name (string)
- Course Level Difficulty (string)
- Required Textbook (string)
- Vector Course List <string>
- Vector Course Videos <mp4>
  Methods
- FindTutor()
- AddTutor()
- EditCourse(string, string)
- AddCourse(string, string)
- DeleteCourse(string)
- courseList()
- setCourseName(string)
- setCourseLevel(string)

SpecialPen Class Attributes

- Pen Tracker (string)
- Vector colorList <string>
- Methods:
- OutputPen()
- UseDifferentColor(colorsList.at(color))
- BoldLetters()
- InsertEquation()

Connector Class Attributes

- Countdown of Duration (int)
- Methods:
- ShowNames()
- RaiseHand()
- WhiteBoard()
- UseChat()
- ConnectStudents()

Interact Class Attributes

- Questions (string)
- Methods:

- ShareScreen()
- RemoteAccess()
- askQuestions(string)

Answer Class Attributes

- Answer (string)
- Methods:
- AnswerQuestion(string)

Attributes Description

User Class

- String attribute 1: This attribute is of the String data type and will be the ID or username of the customer.
- String attribute 2: This attribute will be of the String data type and will be the user's legal name.
- String attribute 3: This attribute will be a String data type and will collect the user's email address for use with their account.
- String attribute 4: This attribute will be of a String data type and will contain the user's password.
- String attribute 5: This attribute will be a string data type that will categorize the account as either a tutor or a customer.

Customer Class

- String attribute 1: This attribute of string data type will contain the user's payment method.
- Int attribute 2: This attribute will be of an int data type and will contain the user customer's age.
- String attribute 3: This attribute will be a string data type and will contain a promo code the customer can enter for special deals.
- String attribute 4: This attribute will be a string that will have the registration date of the user.
- String attribute 5: This attribute will be a string that will contain the rating of the student account.

Payment Class

- Attribute 1: This attribute will have the cost of the payment that is due for the user.
- String attribute 2: This attribute will have the name of the account holder to whom the payment is due for.
- String attribute 3: This attribute will contain an ID that will be used to reference the specific transaction.
- Int attribute 4: This will be an int data type that contains the card number.
- Int attribute 5: An int data type that will contain the expiration date in a certain format.
- Int attribute 6: An int data type that will contain the card's CCV number.
- String attribute 7: A string data type that will contain the cardholder's name.

Course Class

- String Attribute 1: This attribute will contain the course ID in the form of a string.
- String attribute 2: This attribute will contain the course's name in the form of a string.
- String attribute 3: This attribute will contain the difficulty level in a categorical form, data type being a string.
- String attribute 4: Will contain the textbook name for the corresponding course.
- Vector list Attribute 5: An array that contains the courses that are part of the software.
- Vector list Attribute 6: An array that contains the videos corresponding to the courses.

Tutor Class

- String attribute 1: This attribute will contain the subjects that the specific tutor specializes in.
- String attribute 2: This attribute will contain the bank information/ money transfer information that is corresponding with the tutor account.
- String attribute 3: This attribute will contain the rating that is given to the tutor.
- Vector List attribute 4: This will contain the list of all the tutor accounts by name in the form of a string.
- Vector List attribute 5: This will contain an array of strings that will contain the courses that a certain tutor will teach.
- Vector List attribute 6: This will contain an array that holds the dates and times of a specific tutors

Special Pen Class

- Attribute 1: This attribute will contain the tracker for the pen. Will be attributed with the input device the user is using. Ex. Mouse, trackpad, tablet, etc.
- Vector List Attribute 2: This attribute will be of a string type that will contain the color codes for the pen to use.

Connector Class:

- Int attribute 1: This attribute will contain the running minutes of the session.

Interact Class:

- String attribute 1: This string will contain the question that the customer will be asking the tutor.

Answer Class:

- String attribute 1: This string will contain the answer for the specific question that was asked by the customer.

Operations Description

Administrator Class

- manageCourses: This function will allow the administrator to update and change courses within the software.
- manageLanguages: This function will allow the administrator to add and remove string inputs that will categorize different languages that the session will be using.
- manageUsers: This function will allow the administrator to edit user profiles along with managing and deleting them.
- manageTutors: This function will allow the administrator to edit tutor profiles along with deleting them.
- customerSupport: This will allow the administrator to look into customer support questions with questions that they may have inputted. This function may prompt a chat box between the user and the administrator.
- tutoringSupport: This will allow the administrator to look into issues tutors may be having and assist. This function may prompt a chat box to troubleshoot issues that the tutor may be having. Will intake string inputs along with chars for communication.
- paymentSupport: This will allow the administrator to contact 3rd party payment actors to troubleshoot issues they may have. This function will intake strings and chars as a means of communication between both parties.
- manageTutoringTime: This function will open a chart with the different times that tutors will have to host their sessions. Administrators will have the ability to delete values and other data types that the tutor user may input into the chart. Administrators will also have the ability to add to this chart as well, taking in integers to fill in attributes for the times.

User Class

- Login: This function will allow the user to log in. It will intake 2 strings, one for the password and one for the username. This will come in the form of a sign in page. Upon checking the database if the inputs are equivalent, the user will be signed in. Otherwise it will prompt the user to enter information again.
- Logout: This function will allow the user to log out of their account from the software. This will come in the form of a button along the top or bottom edge of the page.
- addnewUser: This function will intake 4 strings that will account for the user's legal name, userID, password, and email address. The interface will be in the form of a new sign up page that will be connected to the login and landing page. This will then add the information to the user database for storage and encrypted to protect user information.

- editAccount: This function will allow the user to edit attributes that are connected to their account. This will take form as a new page where users can see all the attributes inputted and change them as needed.
- setType: This function will set the type of account of the user by a string. When inputted “tutor” the account will be part of the tutor class and inherit attributes from there. Otherwise it is set to the customer class where it will inherit those attributes.
- checkExistingUser: This function will check if the user created account matches with others in the database previously. It will compare strings of the user by looking at the user’s name and userID to see if the user is already in the system.
- setEmail: This is a setter function that will intake a string and set an accounts email address.
- setUsername: This is a setter function that will intake a string and set an account’s password.
- setPassword: This is a setter function that will intake a string and set an account’s password.

Customer Class

- setNewUser: This function will set the user account as a member of the customer class. Will most likely take form as a selection bubble on the signup page.
- editAccount: This function will invoke the editAccount function from the user class.
- checkExistingUser: This function will invoke the checkExistingUser from the user class.
- watchTutoringVideos: This function will come in the form of a selection on the landing page that will allow the customer to traverse the database/library of tutor made videos on different courses and subjects.
- rateTutors: This function will allow the customer to rate a specific tutor that they are connected with. This will intake a string that will categorize the first character as a number rating followed by a comment that will contribute to the tutor. This will take the form of a button selection that will bring up an input box and prompt the user on how to format their rating correctly.
- avoidTutors: This function will allow the customer to add the tutor to a form of a block list so the customer and tutor will not be paired up again in the future. This will simply add the tutors name to the customers block list and if the strings match up on a condition check, they will not be paired. The interface will be in the form of a block button in the tutoring session.
- scheduleTutors: This function will prompt the user to schedule a time to have a session with a specific tutor. This will come in the form of a new page where the customer can browse and filter between courses and availability time, or even sort through a list of tutors. This will intake strings for the filters.

- saveNotes: Customers will be able to save notes that were provided by the tutor during the tutoring session. This will output a pdf and send it to the customer's email linked to their account. This will come in the form of a button press during the tutoring session window.

Tutor Class

- setNewAccount:This function will set the user account as a member of the tutor class. Will most likely take form as a selection bubble on the signup page. No further input needed.
- editAccount: This function will invoke the editAccount function in the user class.
- rateStudents: This function will allow the tutor to rate students based on experience. This will take the form of an input window after a button press to prompt the rate system. This will collect a string that will categorize the first character as a rating and the rest to be a comment following the rating. This string input will be stored in the reviews database connected with the customer.
- avoidStudents: This function will allow the tutor to add students to their own block list in case the customer provided an unpleasant experience. This will add the customer to the tutors list (potentially string array) and if they match in the future, the session will not be allowed to connect and will prompt the reason.
- sendNotesToStudent: This function will save the notes in the tutoring sessions window as a pdf and send it to the customer's email address in the system. This will come in the form of a button press in the tutoring window or as a prompt after the session has ended.
- tutorList: This function is a getter method that will retrieve the list of names associated with the tutor class.
- tutorCourses: This is a getter method that will retrieve the courses that a tutor tutors in.
- addTime: This is a setter method that will add a time and date that the tutor is willing to tutor in.
- addCourse: This is a setter method that will add a course to the tutors course list.
- removeCourse: This method will delete a course from a tutors' course list. It will intake a string datatype.

Payment Class

- getPaymentStatus: This function will allow customers to check the status of their payments. Most likely there will be a drop down selection with the payment ID upon the payment page and will only be able to view the status provided by the 3rd party payment processor and no edits will be allowed.
- addCard: This method will add a card to the users account and will intake 3 strings and a string as the parameters.
- deleteCard: This method will delete a card from a user's account.

- setCardNumber: This is a setter method that will intake a string and set the card number.
- setExpDate: This is a setter method that will intake an int and format it to set as the expiration date of a card.
- setCCV: This is a setter method that will set the cards CCV number and will intake an int data type.

Course Class

- findTutor: This function will find tutors that will be able to tutor a certain course. This will intake a string and compare with the list of tutors within the system. From here the customer will be able to schedule an appointment.
- addTutor: This function will add tutor accounts to the list of applicable tutors able to assist in the course. This will intake a string and add it to the list (array) of applicable tutors. This function will most likely be accessible to the administrator and tutor classes and will allow them to edit the list. The function will take the form of a input box viewable to the administrator and tutor classes to add the tutors to the list.
- editCourses: This function will allow administrators to edit the course details and attributes of the courses. This will take the form of a new page where the courses will be displayed and the attributes can be edited via clicking on the data and changing them.
- addCourses: This function will allow the administrator to add courses to the system. This will intake 4 strings to provide a new course ID, name, and level, along with the textbook name suitable for the course. This will take form as a drop down menu in the courses page accessible by the administrator.
- deleteCourses: This function will allow the administrator to delete courses from the system. This will take the form of a delete button next to the individual courses displayed in the courses page. The administrator will be the only account type with this privilege.
- courseList: This is a getter method that will retrieve all the courses that the software will have available.
- setCourseName: This is a setter method that will set the course name. It will intake a string and use it to set the name.
- setCourseLevel: This is a setter method that will set the course level. It will intake a string that will be used to categorize the difficulty of the course.

Special Pen Class

- outputPen: This function will work with the input device to provide a way for the pen to be visible on the virtual whiteboard. It will create a line whenever the input device is dragging and clicking inputs are being read.

- useDifferentColor: This function will change the color of the output coming from the pen. Different colors will be selected via a drop down menu and the tutor or customer will be able to select which they would like to use.
- boldLetters: This function will embolden the output of the pen. The user will be able to highlight parts of the virtual board and make the trace path thicker for the trails that are within the highlighted area. This function will be activated with a button press on the tutoring session window.
- insertEquations: This function will allow users to type and input mathematical equations on the note sheet or whiteboard. This will intake char and string inputs from the users and will be activated by a button press in the tutoring session window.

Connector Class

- showNames: This function will display the names of the users connected to the tutoring session. This will not take an input but will display the names of all users via a pop up window when pressed by the button in the tutoring session window.
- raiseHand: This function will display a hand connected with the customers name to the tutor to indicate they are raising their hand. This will be activated by a button press on the tutoring window through the customer side.
- useWhiteBoard: This function will launch the virtual whiteboard. This function will expand the window of the tutoring session so users can draw, write or type on this interface. The function will be activated by the user in the form of a button along the tutoring session window.
- useChat: This function will launch a separate mini window displaying 2 boxes. One for the chat output and one for the user input. This will intake strings and chars from the user and the message will be displayed along the output window with their name by the message. This function will be activated via button press within the tutoring session window.
- connectsStudents: This function will allow the tutor to grant access to students in the waiting area to the tutoring session. This will come in the form of a window before the tutoring session and will have buttons accompanying the students name to deny or accept them into the session.

Interact Class

- sharesScreen: This will allow the customer or tutor to share their computer screen. This will display the user's computer screen and will be activated by a button on the tutoring window. This will prompt the host and the host will allow access to the one who activated the function.
- allowRemoteAccess: This function will allow the customer to give computer access to the tutor. This will only be accessible when shareScreen is active. This will be in the form of a button press along the tutoring session window.

- askQuestions: This function will allow the customer to ask the tutor a question. This function will prompt an input box for the customer to send to the tutor. This message will be sent to the tutor to invoke the answerQuestion function. This will be activated by a button press in the chat window. This will intake strings from the customer.

Answer Class

- answerQuestion(): This function will allow the tutor to answer private questions from the customers. The question from the customer will be displayed and an input box will pop up to prompt the tutor to enter the answer to the question from the customer. This will intake strings from the user.

Test Plan 1:

- For the first test plan we will be targeting the features of simply creating an account and logging in then logging out. On the unit level, we will test the setter methods and check that they will intake inputs that are valid for the software. This will be tested by an EQ balance method which will randomly select inputs from different test classes.On the functional level we will be testing the addNewUser method which will integrate the setter methods and create a new user account. The system level will be tested by running the cycle as a whole which would be to test the account creation and then test the login methods and finish off by checking the logout method.

Unit Test Plan:

- These are the test classes that we created to test the inputs for the setter methods that will set the user information.

Class Number	Test Case	Condition
1	Valid Username Input (6 char min)	userName >= 6
2	Invalid Username Input	string != "username", string == " username"
3	Blank Username Input	"", " ", " ", nullptr
4	Valid Email (must have @gmail or something)	string == "EMAIL" contains '@'
5	Invalid Email	string != "EMAIL", no '@' included
6	Blank Email	"", " ", " ", nullptr
7	Valid Password (atlest 8 chars)	validPassword >= 8 chars
8	Invalid Password	validPassword < 8, string && validPassword != "actual assword"
9	Blank Password	"", " ", " ", nullptr
10	Valid Age (5-95)	valAge > 5 && valAge <95
11	Invalid Age	valAge < 5, valAge > 95, string "32432dsfs",string "@#Fd"
12	Blank Age	"", " ", " ", nullptr

Functional Test Plan:

- The functional level will be tested by creating different test cases and seeing whether they will produce a valid response within the system. It will involve integrating the setter methods into the addNewUser method.

Account Testing

Test Case 1: VALID Username = Ra32498, Email = ra32498@gmail.com
Password = abcdefghi, Age 25

- The user name Ra32498 goes through the 1st class and it passes the conditions of being more than 8 characters.
- The email ra32498@gmail.com goes through the 4th class, and it passes the condition for having "@" in the email.
- The password goes through 7th class to check if the password is at least 8 characters long.
- The age of 25 goes through the age class, and it passes the condition since 25 is bigger than 4 but less than 25
-

Test Case 2: INVALID Username = sam619, Email = icereamboy@gmai.com Password = " ", Age 4

- The user name sam619 gets rejected immediately in the beginning since the user name did not pass the condition of having the right amount of character for the user name. The test cases immediately ends at the first class

Test Case 3: VALID Username = JustinLee, Email = jl@sdusu.edu/
Password = dgfgd3333, Age 30

- The user name JustinLee goes through the 1st class and it passes the conditions of being more than 8 characters.
- The email jl@sdusu.edu goes through the 4th class, and it passes the condition for having "@" in the email.
- The password goes through 7th class to check if the password is at least 8 characters long. The password passes the conditions
- The age of 30 goes through the age class, and it passes the condition since 30 is bigger than 4 but less than 25

Test Case 4: INVALID Username = AndrewLIM, Email= andrew@gmail.com, Password = jgj, Age 45

- The user name AndrewLIM goes through the 1st class and it passes the conditions of being more than 8 characters.

- The andrew@gmail.comgoes through the 4th class, and it passes the condition for having "@" in the email.
- The password goes through 7th class to check if the password is at least 8 characters long. The password does not fit the criteria since the password is only 3 letters long so this make this test case invalid

Test Case 5: VALID Username = JasonChen33, Email=JasonChen@yahoo.com
Password = sdfdsfdsfds, Age 32

- The user name JasonChen33 goes through the 1st class and it passes the conditions of being more than 8 characters.
- The andrew@gmail.comgoes through the 4th class, and it passes the condition for having "@" in the email.
- The password goes through 7th class to check if the password is at least 8 characters long. The password does fit the criteria since the password is only 11 letters long so this make this test case valid
- The age of 32 goes through the age class, and it passes the condition since 32 is bigger than 4 but less than 25

Test Case 6: INVALID Username = Pete, Email=PeterDim@sdsu.edu
Password = hhhhhhhhhh, Age 21

- The user name Pete goes through the 1st class and it does not pass the conditions of being more than 8 characters. This test fails immediately. This validates that the test is invalid.

System Test Plan:

- Similarly to the functional level testing. This will now integrate more the system as a whole by implementing the login method to check the info inputted across the database and checking if it matches with the information that is already present. We will assume that the logout method will be successful and just check the integration of the methods.

Test Case 1: VALID Username = Ra32498, Email = ra32498@gmail.com
Password = abcdefghi, Age 25

- The account will be created with the information provided above and from the previous test case we know that this will succeed.
- Now the account is created we will attempt to login. The login information used will be: Username = Ra32498, Password = abcdefghi.
- The login method will check whether the information entered will match the information in the database.
- In this case it matches properly therefore letting the user login.

Test Case 2: INVALID Username = Ra32498, Email = ra32498@gmail.com
Password = abcdefghi, Age 25

- The account will be created with the information provided above and from the previous test case we know that this will succeed.
- Now the account is created we will attempt to login. The login information used will be: Username = Ra32948, Password = abc.
- The login method will check whether the information entered will match the information in the database.
- For this certain case it is found that the username and password do not match the information within the database leading as well as the password not meeting the required string size, therefore causing an error and prompting the user to input a valid password.

Test Plan 2:

- The features targeted for this test plan will be to test the payment class. On the unit level, we will be testing the valid inputs for credit card information and seeing whether the methods will intake information within the correct format. On the functional level we will be testing the addCreditCard method to test the integration of the method with the setters of the class. We will be testing the system level by logging into an account and adding a credit card to the account.

Unit Test Plan:

13	Valid Card Number (16 Digits)	int validCardNumber == 16 ints
14	Invalid CardNumber	int cardNumber != "", " ", " ", nullptr || int cardNumber <= 16
15	Blank Card Number	int blankCardNumber == "", " ", " ", nullptr
16	Valid Exp Date (format: 0324 = 03/24)	int validExpDate length == 4 || int validExpDate length == 5 and expDate [2] '/ '
17	Invalid Exp Date	int invalidExpDate != "", " ", " ", nullptr || int invalidExpDate < 4 or < 5 || int inValidExpDate =< 1299 and 12/99
18	Blank Exp Date	int blankExpDate == "", " ", " ", nullptr
19	Valid CCV (3 digits	int validCCV == 3
20	Invalid CCV	int validCCV != "", " ", " ", nullptr || int CCV < 3
21	BlankCCV	int blankCCV == "", " ", " ", nullptr
22	Valid Name (no numbers allowed)	string validName == const char || string validName != 'A-Z' || string validName != 'a-z'
23	Invalid Name	string invalidName == "123" || string invalidName == "A2C" || string invalidName ="1B3"
24	Blank Name	string blankName == "", " ", " ", nullptr

Functional Test Plan:

- To test the software along this path on the functional level, we will be testing the addCard method to check the integration of the setter methods within this class.

Payment Testing

Test Case 1: VALID

- CardNumber = 4413 2826 5213 4316, Expiration Date 11/23 or 1123, CCV = 611, Name = Justin Lee
  - CardNumber passes the condition when CardNumber is equal to 16 integers. Expiration passes conditions when meeting the format 11/23 with 4 integers plus 1 special character or format 1123 that is equal to 4 integers. CCV passes the condition when equal to 3 integers. Name contains letters only with a space in between first and last name, this would conclude the valid test case.

Test Case 2: INVALID

- CardNumber = 4413 2826 431, Expiration Date 11/2 or 112, CCV = 61, Name = Yeti H20
  - CardNumber fails the condition with 15 integers, should be equal to 16 integers. Expiration fails the condition with 3 integers plus 1 special character instead of 4 integers plus 1 special character or 112 with 3 integers instead of 4 integers. CCV 61 is 2 integers and should be equal to 3 integers. Name fails the condition when the last name contains an integer and should contain letters only, this would conclude the invalid test case.

Test Case 3: VALID

- CardNumber = 4480212358975465, Expiration Date 12 / 25 or 1 2 2 5, CCV = 3 1 2, Name = jAsoN cHeN
  - CardNumber passes conditions with 16 integers without spaces in between. Expiration Date 12 / 25 ir 1 2 2 5 still passes conditions with random space in between each integer or special character. Name passes condition with random lower and upper letters, this would conclude the valid test case.

Test Case 4: INVALID

- CardNumber = "", " ", ", Expiration "", " ", ", CCV = "", " ", ", Name = "", " ", "
  - CardNumber fails the condition with blank spaces, there are no integers that equal 16 integers. Expiration date fails the condition with blank space, there are no integers that equal to 4 integers plus 1 special character or 4 integers. CCV fails the condition with blank space since there are no integers that equal 3 integers. Name fails the condition with blanks space since there is no entry with letters only, this would conclude the valid test case.

Test Case 5: VALID

- CardNumber = 44 561232 7524, Expiration 10/ 25 or 1 025, CCV = 2 12, Name = justin Lee
    - CardNumber passes the condition when CardNumber is equal to 16 integers even with random space in between. Expiration Date passes the condition with random spaces in between the format 10/ 25 or 1 025. CCV passes a condition with 3 integers with random space. Name passes conditions with random lower upper case format, this would conclude the valid test case.

Test Case 6: INVALID

- CardNumber = 445612327ABC, Expiration AA/26 or AA26, CCV = A57, Name = Justin 123
    - CardNumber fails the condition with 17 integers, should be equal to 16 integers. Expiration Date fails the condition with letters as month and an additional integer making it greater than 4 integers plus 1 special character. CCV fails the condition with 4 integers, should be equal to 3 integers. Name fails the condition with blank space entry for name, this concludes the invalid test case.

Test Case 7: INVALID

- CardNumber = 4497 1123 47856571, Expiration Date 1 2 / 2 35 or 1 2 2 35, CCV = 5 7 2, Name = 123 321
    - CardNumber fails the condition with 17 integers instead of 16 integers. Expiration Date fails the condition with 5 integers plus 1 special character, should be 4 integers plus 1 special character and 5 integers should be 4 integers. CCV fails the condition with 4 integers instead of 3 integers. Name fails the condition with integers as the entry, should be letters only, this concludes the invalid test case.

System Test Plan:

- On the system level we will be testing this by attempting to login and then adding a card to the account. This will check the getters and check the input data across with what is in the database. It will then add a card associated with the account to the payment database as well.

Test Case 1: Username = Ra32498, Password = abcdefghi

- We will test the login with the information provided, in previous tests we know that this input will be valid.
- Next we will test the addCard method with the information: CardNumber = 4413 2826 5213 4316, Expiration Date 11/23 or 1123, CCV = 611, Name = Justin Lee.

- The method will check whether the inputs are valid and if they are the card will be added. In this case, this is valid.

Test Case 2: Username = Ra32498, Password = abcdefghi

- The login information will be tested once again in this test case. This will be successful.
- Next we will be testing the addCard method with the information: CardNumber = 4413 2826 431, Expiration Date 11/2 or 112, CCV = 61, Name = Yeti H20.
- The card information provided does not follow the bounds of the input fields therefore this test case will be rejected and marked as invalid.