Student-Teacher Response System

Group 3 Alonzo Altamirano, Becca Arraya, Jieun Park, Shengjie Zhai, and Taylor Santos

Project Plan

1. Purpose and Audience

    i. The intended audience:
        a. This document is intended for developers and stakeholders of the Student-Teacher Response System
    ii. Purpose:
        a. To review team roles and responsibilities for a team member
        b. To provide risk and mitigation
        c. To document development process and techniques that will be implemented
        d. To list project schedule and milestones for team member verification

2. Project Background

    We are creating a student/teacher response system allowing direct communication between a student and a teacher. A teacher can ask multiple choice questions with answer possibilites in the teacher application that students can answer in the students application. Also, students can ask a question and have the text sent to the teacher directly.

    Our project uses the python tkinter library for both the student and teacher user interface. To make sure students app can connect with the teachers application, we catch a TCP/IP address by a using web server. The quizzes can be created ahead of time and saved on the teacher's computer. During a lecture, the teacher may load any previously created quizzes to be sent to the students.

3. Team Roles and Responsibilities

Role	**Team Member**	**Responsibilities**
System Architect	Everyone	Specify designs that maintain accordance with system requirements with which modules will be implemented. Communicate with implementers in order to exact and refine

		designs.
Requirements Analyst	Everyone	Explore and specify objectively verifiable requirements of system and software. Communicate with system architects in order to exact and refine requirements.
Quality Control	Becca	Develop testing plans(procedures) and perform integration tests.
Developer(Back End)	Taylor, Alonzo	Realize(implement) designs of logical modules in accordance with specified requirements. Produce documentation for assisting developers and maintainers to utilize and modify the code.
Developer(Front End)	Shengjie, Jieun	Realize(implement) designs of user interface modules in accordance with specified requirements. Produce documentation for assisting developers and maintainers to utilize and modify the code.
Technical Writer	Becca	Produce user documentation. Produce user manual. Maintain logs of group events.
Configuration Control	Alonzo	Manage builds and modifications to the system. Facilitate usage of version control for all group members.
Project Management	Becca, Shengjie	Develop a Project Plan. Maintain progress of system development within project schedule. Facilitate communication within group.
Tester	Alonzo, Shengjie, Jieun, Taylor	Develop and perform unit tests on modules of the system.

4. Risks and Mitigation

Risk	**Solution**
Tkinter becomes unusable for our	

project(a critical bug is discovered, Tkinter is deleted and support is discontinued, etc).	We will pursue usage of another GUI library to rely on from the pool of comparable technologies available(pyQT, FLTK, etc).
We get behind schedule and the project is due soon.	We will cut out any advanced functionality causing delays and focus on only the core function. We will increase the hours that each of us devote to the project and if need be meet on additional occasions.
Someone gets sick or dies.	We will document our work and decisions not only to assist in future maintenance but also to account for this possibility. By having multiple system architects, managers, and developers with knowledge of the same topics we will develop redundancy for this proposed event.
The current network configurations does not allow for a host (the professor).	We will create the ability to host the server remotely.
There is a lack of communication within the group about milestones reached and further steps in the development of the software.	We will use Github with mulitple branches to upload code and Slack for communication between members. A Gantt chart will be used to communicate responsibilities and timelines.

5. Process

The development process will use an iterative approach, meaning we will go through the important steps in the software lifecycle in iterations. These main steps include requirements analysis, design, testing, and evaluation. Since we do not have active customers to work with, our process is not agile but a series of waterfalls. During each group meeting, we will start by going over the requirements making sure these are met and making some changes small changes along the way. For the main design, the workload is broken up by modules with the respective team members delegated for each module development. The back-end components (database and network) and the front-end components (student/teacher interface) will be developed seperately and merged in the end to complete the software. Knowledge of the functionality of both components will

need to be known by at least one member from each component, since merging must be implemented. Testing of the software will occur simulatenously with the design, since the coders will be testing while creating the base product. Quality control is kept up throughout the project to mitigate risks, ensure requirements are met, and allow for added features if time allows. Meetings will be held two times a week for quality control purposes and to keep the milestone timeline on pace. Any project risks will be discussed during project meetings to avoid risk accumulation.

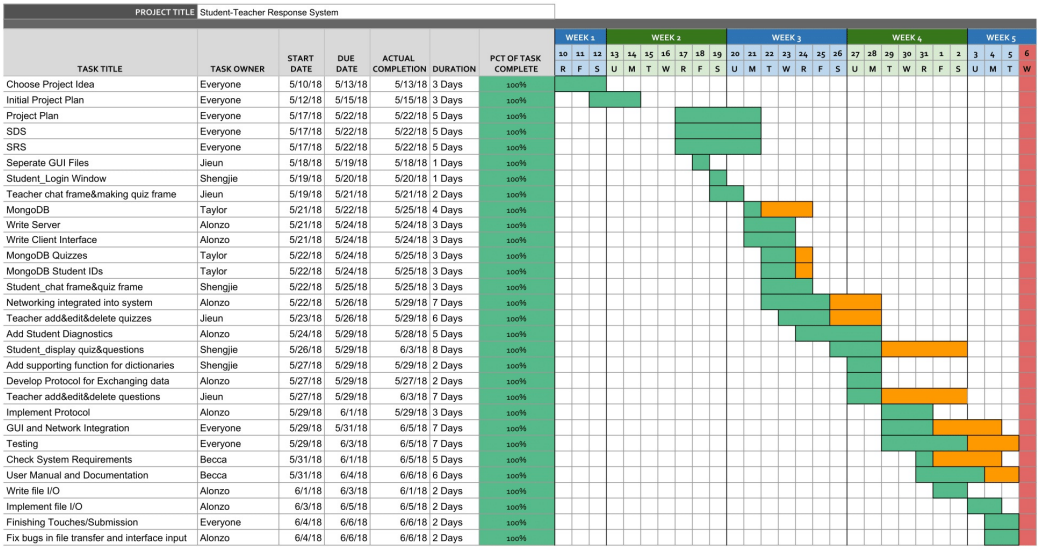
6. Mechanisms, Methods, Techniques

   The software will use the Tkinter library of Python3 to develop the graphical user interface. The quizzes will be stored on individual text files generated by the application. The network will be written in Python3 with a web server that can make a connection between the student and teacher applications. The web server will ensure that student can directly send information to the teacher by asking a direction question or answering a quiz question sent via the teacher application. The TCP/IP protocol enables the student and teacher applications to connect with the server.

7. Detailed Schedule and Milestones

Milestone	**Task Description**	**Task Owner**	**Start Date**	**Due Date**
1	Choose Project/Explore Technologies, specifically database, programming language, network protocol	Everyone	Thursday 05/10/18	Sunday 05/13/18
2	Write Project Plan, SRS, SDS Complete	Everyone	Tuesday 05/15/18	Tuesday 05/22/18
3	Create files in Python to store data in MongoDB	Taylor	Friday 05/15/18	Thursday 05/24/18

4	Create files for Network in Python	Alonzo	Thursday 05/10/18	Saturday 05/26/18
5	Create both student/teacher GUI view	Jieun/Shengjie	Saturday 05/19/18	Tuesday 05/29/18
6	Integrate GUI and Network together to allow student and teacher to interact	Everyone	Monday 05/28/18	Thursday 05/31/18
7	Go through quality assurance plan and requirements to make sure use cases and requirements are met	Everyone	Friday 05/25/18	Friday 06/01/18
8	Go through SRS to make sure requirements are met and update SRS as needed	Becca	Tuesday 06/06/18	Sunday 06/03/18
10	Write User documentation and User Manual	Becca	Tuesday 06/05/18	Monday 06/04/18
11	Application Easily Accessible and Submission	Everyone	Wednesday 06/06/18	Wednesday 06/06/18



PROJECT TITLE	Student-Teacher Response System

TASK TITLE	TASK OWNER	START DATE	DUE DATE	ACTUAL COMPLETION	DURATION	PCT OF TASK COMPLETE
Choose Project Idea	Everyone	5/10/18	5/13/18	5/13/18	3 Days	100%
Initial Project Plan	Everyone	5/12/18	5/15/18	5/15/18	3 Days	100%
Project Plan	Everyone	5/17/18	5/22/18	5/22/18	5 Days	100%
SDS	Everyone	5/17/18	5/22/18	5/22/18	5 Days	100%
SRS	Everyone	5/17/18	5/22/18	5/22/18	5 Days	100%
Seperate GUI Files	Jieun	5/18/18	5/19/18	5/18/18	1 Days	100%
Student_Login Window	Shengjie	5/19/18	5/20/18	5/20/18	1 Days	100%
Teacher chat frame&making quiz frame	Jieun	5/19/18	5/21/18	5/21/18	2 Days	100%
MongoDB	Taylor	5/21/18	5/22/18	5/25/18	4 Days	100%
Write Server	Alonzo	5/21/18	5/24/18	5/24/18	3 Days	100%
Write Client Interface	Alonzo	5/21/18	5/24/18	5/24/18	3 Days	100%
MongoDB Quizzes	Taylor	5/22/18	5/24/18	5/25/18	3 Days	100%
MongoDB Student IDs	Taylor	5/22/18	5/24/18	5/25/18	3 Days	100%
Student_chat frame&quiz frame	Shengjie	5/22/18	5/25/18	5/25/18	3 Days	100%
Networking integrated into system	Alonzo	5/22/18	5/26/18	5/29/18	7 Days	100%
Teacher add&edit&delete quizzes	Jieun	5/23/18	5/26/18	5/29/18	6 Days	100%
Add Student Diagnostics	Alonzo	5/24/18	5/29/18	5/28/18	5 Days	100%
Student_display quiz&questions	Shengjie	5/26/18	5/29/18	6/3/18	8 Days	100%
Add supporting function for dictionaries	Shengjie	5/27/18	5/29/18	5/29/18	2 Days	100%
Develop Protocol for Exchanging data	Alonzo	5/27/18	5/29/18	5/27/18	2 Days	100%
Teacher add&edit&delete questions	Jieun	5/27/18	5/29/18	6/3/18	7 Days	100%
Implement Protocol	Alonzo	5/29/18	6/1/18	5/29/18	3 Days	100%
GUI and Network Integration	Everyone	5/29/18	5/31/18	6/5/18	7 Days	100%
Testing	Everyone	5/29/18	6/3/18	6/5/18	7 Days	100%
Check System Requirements	Becca	5/31/18	6/1/18	6/5/18	5 Days	100%
User Manual and Documentation	Becca	5/31/18	6/4/18	6/6/18	6 Days	100%
Write file I/O	Alonzo	6/1/18	6/3/18	6/1/18	2 Days	100%
Implement file I/O	Alonzo	6/3/18	6/5/18	6/5/18	2 Days	100%
Finishing Touches/Submission	Everyone	6/4/18	6/6/18	6/6/18	2 Days	100%
Fix bugs in file transfer and interface input	Alonzo	6/4/18	6/6/18	6/6/18	2 Days	100%

Meeting Notes

Task/Deliverable	Asignee	Start Date	End Date	Estimated Time
Choose project idea	Alonzo	May 10th	May 13th	4 hours
Choose programming language	Alonzo	May 10th	May 10th	20 minutes
Choose GUI library	Shengjie, Jieun	May 10th	May 10th	4 hours
Write initial project plan	Everyone	May 12th	May 14th	3 hours
Write project plan, SDS, SRS	Everyone	May 17th and May 21st	May 21st	2 days

Discussing data format for GUI integration	Alonzo, Jieun, Shengjie	May 28th	May 28th	2 hours
Check System Requirements	Everyone	May 31st	May 31st	2 hours
Merging Network and GUI	Alonzo, Jieun, Shengjie	June 3rd	June 3rd	6 hours
Continue Merging Network and GUI, Writing documents	Everyone	June 5th	June 5th	10 hours
Project Completion.	Everyone	June 6th	June 6th	6 hours

Software Requirements

1. Introduction

    i. Intended Audience and Purpose
    - Users - This document describes the functionality of our product, including use cases and external behavior. A user looking to understand the capabilities of our product should consult this document.
    - Marketing - This document will serve as a reference for any marketing materials produced for the product. The product's description from the perspective of a potential user is given in the Concept of Operations section, which can be referenced when producing advertising materials.
    - Product Management - A high-level, non-technical overview of the functionality of the product is necessary for managing the distribution of personnel and resources, as well as for gauging progress towards project completion. The features described in this section can be used as a starting point for laying out project milestones.

    ii. How to Use the Document
    - Section 2 contains the Concept of Operations. It describes the context in which the software will be run, as well as the third party tools, libraries, and APIs used by the software. Use Cases are listed, detailing common scenarios the user will encounter when using the product and the desired product behavior in each scenario.
    - Section 3 contains the Behavioral Requirements. The inputs and their restrictions are listed, as well as outputs. The externally visible outputs as a function of

inputs are described.

2. Concept of Operations
   The purpose of the software is to provide a tool for professors and students to ask questions and receive instant feedback. We expect the application to be run in a large-scale classroom where both the professors and students will launch the application on a personal computer/laptop. Professors will be able to ask multiple-choice questions and receive any comments or questions a student sends direclty. Students will be given the opportunity to ask questions about assignments and course material with a chat feature to send to the professor during the class. The student may also receive quiz questions and submit their multiple-choice answer. It enables students to learn from questions others have and helps the teacher glean how the class responds to their lecture material. Although the chat feature may be a distraction during class for the teacher, we imagine the teacher spending time at the end of class scrolling through comments/questions or having an assistant manage messages during the classroom time. Also, a teacher may create quizzes prior to class and pull up saved quizzes to share with students to avoid wasting time making multiple questions during class as well.

   i. System Context

      a. tkinter will be used to as the GUI.
      b. The program is compatible with a version of Python 3.6.5 or newer.

   ii. System Capabilities

      Use Case #1: Initialization

      a. Description: The students wish to connect to the professor's client to begin using the software.
      b. Actors:
         a. One professor
         b. At least one student
      c. Preconditions:
         a. The professor must have the host application installed.
         b. All students must have the client application installed.
         c. All actors must be connected to the same local network.
      d. Flow of Events:
         a. Basic Flow
         b. The professor starts up the host application.
         c. The necessary connection information is displayed on the professor's screen.
         d. The professor shares this information with the students.
         e. The students start up the client applications.

f. The client application prompts the student for their student ID number.
g. The student inputs their ID number.
h. The client application prompts the student for connection information.
i. The student inputs the connection information given to them by the professor.
j. Connection status is displayed on the student's screen.
k. The number of connected students is displayed on the professor's screen.
l. The students and the professor will be brought to their respective main pages

e. Exceptions:
    a. Unable to connect:
    If after step 'h' the client is unable to connect to the professor, the student will be alerted about the connection failure, and will be given a list of potential remedies (Restart the application, verify correct spelling, ensure connectivity to the correct network, etc.) The use case will resume on step 'g'.

Use Case #2: Professor creates a quiz

a. Description: The professor wishes to send multiple-choice question/questions to the students and receive back responses.
b. Actors:
    a. One professor
    b. At least one student
c. Preconditions:
    a. The actors must have gone through the Initialization use case with no errors.
d. Flow of Events:
    a. Basic Flow
        a. The professor starts out on the main page.
        b. The professor selects the option to create question.
        c. The professor inputs the multiple choice answers as well as selecting correct answer.
        d. The professor will submit the question and may create more questions if wanted by repeating steps 'b' and 'c'.
        e. The professor will indicate they are finished editing the quiz.
        f. The quiz will be saved, and the professor will be returned to the main page.
    b. Alternative Flows:
        a. The professor wishes to cancel the quiz creation process:
        Exiting the quiz creation section without publishing or saving will delete the current quiz. This returns the use case to step 'a'.
    c. Exceptions:

            a. A question is not inputted into the application:
               The professor will not be given the ability to save a quiz without first inputting a question. The use case resumes at step 'c'.

Use Case #3: Students receive a quiz

a. Description: The professor has published a quiz, and a student wishes to submit a response.
b. Actors:
    a. Student
c. Preconditions:
    a. The student must have the application running and be successfully connected to the professor.
    b. A quiz must be posted by the professor
d. Flow of Events:
    a. Basic Flow
        a. The student starts out on the main page
        b. The student will be see the quiz name in view.
        c. The student selects the quiz they wish to respond to.
        d. The question and possible answers are displayed to the student.
        e. The student selects the answer of their choice.
        f. The student may change their selection by repeating step 'e'.
        g. Once the student has decided on a final answer, they are given the option to confirm their choice.
        h. The student will be returned to the main page.
    b. Alternative Flows
        a. The student wishes to exit the quiz without confirming an answer:
           The student may exit the quiz at any time, and be returned to step 'c'.
           Their selected but unconfirmed answer will not be sent to the professor.
    c. Exceptions
        a. An answer is not selected:
           The student will be unable to confirm an answer without first selecting one. This will leave them at step 'f'.

Use Case #4: The professor wishes to edit an existing quiz

a. Description: The professor wishes to edit an existing quiz.
b. Actors:
    a. Professor
c. Preconditions:
    a. The professor must have a quiz saved.
d. Flow of Events:
    a. Basic Flow

a. The professor starts on the main page.
b. All saved quizzes are displayed.
c. The professor selects the quiz they wish to edit.
d. Information about the quiz is displayed.
e. The professor is given the ability to modify the quiz name and description.
f. The professor selects one of the quiz's questions.
g. The professor is given the ability to modify the question's answers.
h. The professor selects the option to save their changes.
i. The professor will be returned to the main page.

b. Alternative Flows
None

Use Case #5: A student wishes to send a message to the professor.

a. Description: Students may wish to write down a question to be addressed by the professor at a later time, without disrupting the class.
b. Actors:
a. A student
b. A professor
c. Preconditions:
a. The professor must have the host application running.
b. The student must have the client application running.
c. The student must be on the same network as the professor.
d. The student must have successfully connected to the professor.
d. Flow of Events:
a. Basic Flow:
a. The student starts on the main page.
b. The student inputs their message for the professor into designated message area.
c. The student submits their message.
d. The message appears on the teacher view.
b. Exceptions:
a. The question box is left blank:
The student will not be able to submit without entering a message.

3. Behavioral Requirements

i. System Inputs and Outputs
a. Inputs
a. Student
ID: A string between 1 and 3000 characters in length used to uniquely identify each student.

IP Address: The professor's local IP, used by the students to connect to the T/A client.
Question: A string between 1 and 3000 characters in length that can be sent from a student to the professor.
Quiz Selection: Each available quiz can be selected to give the student's responses.
Quiz Answer: A series of radio buttons indicating the student's selection in a quiz question.

b. Teacher
Select Quiz: Each saved quiz can be selected, indicating it is to be acted on by the "edit", "delete", or "send" functions.
New Quiz: Strings between 1 and 3000 characters in length for the name and description of the quiz. Each question takes strings between 1 and 3000 characters in length for the question and for each of the 5 responses. Each question has 5 radio buttons to select the correct answer.
Edit Quiz: A button to bring up the editing interface for the selected quiz.
Delete Quiz: A button to delete the selected quiz.
Send Quiz: A button to send the selected quiz to the students.

b. Outputs
a. Student
Available Quizzes: A list of all sent quizzes that the student may participate in.
Questions: Once a quiz is selected, a list of questions is displayed.
Answers: Once a question is selected, a list of answer choices is displayed.

b. Teacher
Questions: Questions sent by the students are displayed on the teacher's main page.
Quizzes: A list of saved quizzes is displayed on the teacher's main page.

4. Quality Requirements

Security

- Users need to access application by input student ID in login interface
- Students send message to teacher through server

Reliability

- Student version application can serve multiple users
- Socket update question that teacher assigned should be in real time
- Socket update message that student send to teacher should be in real time

Accessibility

- Users are easy to use the application and each part of functionality is clear for users to locate

5. Expected Subsets
   Our system will be composed of two sections: A student client and a teacher client. The student client is composed of two primary modules: a networking module, and an interface module. The teacher client is composed of three primary modules: a server module, an interface module, and a file I/O module. The student's networking module handles sending and receiving data from the teacher's server module. Both interface modules handle user interaction. The teacher's file I/O module handles saving and loading quizzes to the filesystem.
   The security requirement applies to our networking modules, but not the interface module. As long as the network connection between the student and professor is secure, the student's interface does not need to fulfill a security requirement. Due to the storage of quizzes in local text files on the professor's computer, the securing of the file I/O module relies on the security of the professor's computer. Anyone with access to the professor's personal files will have access to the saved quizzes.
   The reliability requirement must be met with the network module and the interface module. The application must serve multiple users through the network. The reliability of the application is limited by the reliability of the local network. Therefore, a high-quality network connection is recommended for both the professor and the students.

6. Fundamental Assumptions
   The software is developed using Python 3.6.5; therefore the user must have this version of Python or newer. Furthermore, the user must have an accessible network signal through the entire use of the application. If the network signal is lost, the user will not have access to the full functionality of the software. The students and teachers will be on the same network for interactions. If the students are not on the same network, there will not be any communications between teacher and student.

7. Apendices

   i. Definitions and Acronyms

      a. Definitions
      b. Acronyms and Abbreviations
         a. GUI: graphical user interface
         b. SRS: software requirements specification
         c. SDS: software design specification
         d. TCP/IP: transmisson control protocol/internet protocol

   ii. References

a. https://docs.python.org/3/library/tk.html
b. https://www.python.org/downloads/

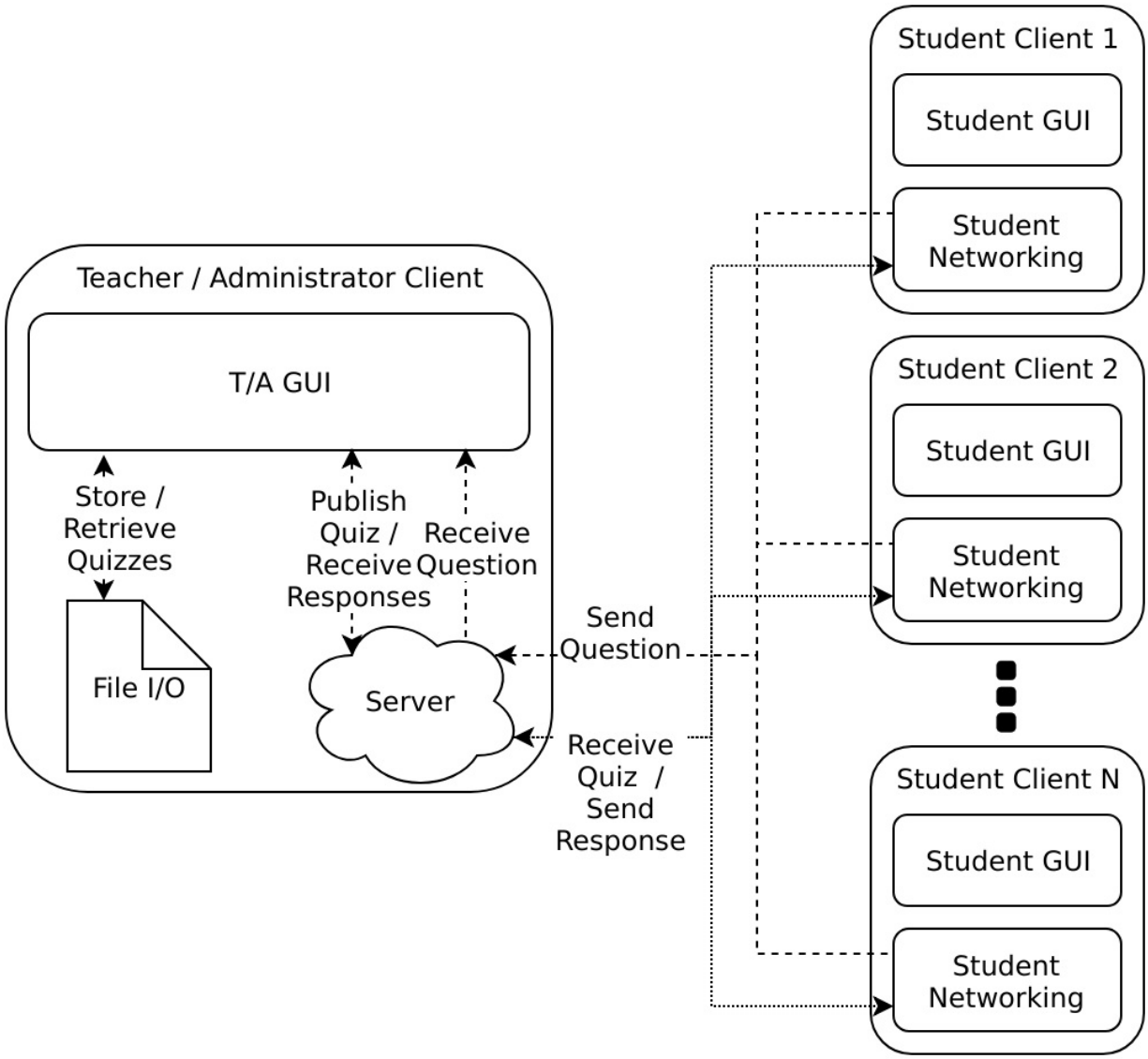
Software Design

Software Architecture

Our system will have two separate user interfaces: one for the student client, and one for the the T/A (teacher/administrator) client. This is done to clearly divide the functionality of both and provide greater cohesion between modules associated with a student and modules associated with the professor. The system will provide distinct functionality depending on whether a user is a student or a teacher. A teacher does not need to connect to another teacher and a student does not need to connect to another student, so the behavior of each should be independent of each other.

A server module is incorporated into the T/A client. The teacher is essential for hosting a class discussion warranting feedback. Without the teacher, there is no reason for students to use the system, thus the T/A client and the server have a functional cohesion, to serve the student. Each student client will have a networking module which handles connection and communiaction with the T/A server. We have chosen to develop the server as a separate module to allow for future maintenance. Should we decide to separate the server from the T/A client, it will be easier due to the module's limited coupling by design.

A database will be incorporated into the T/A client in order to store premade quizes and a whitelist of student IDs. Quizzes made by the professor in the T/A client can be sent to the database module to be saved, allowing for later retrieval. The professor will also be able to create a student ID whitelist, which will prevent unauthorized students from participating in the class. This whitelist will be stored in the database.



Detailed Design

To acheive manifestation of our system, we will have three main components. Client Applications, a Server, and a File I/O.
The Clients and Server will communicate with each other using Internet Protocols. The professor client writes to the file system through File I/O.
GUI:
The user interface will display the student or professor data. The user interface will provide methods for sending messages, sending answer responses for the student and for creating quizzes, sending quizzes, and recieving messages for the teacher. Both the student and professor interface are components of their own respective clients.
Network Interface:
The network will handle all of the logical functionality of our client along with the network communication. The network will have a method for initializing connection with the server which at that point will allow it to spawn threads of processing to communicate with the Server. These threads will carry methods for recieving communications, for sending

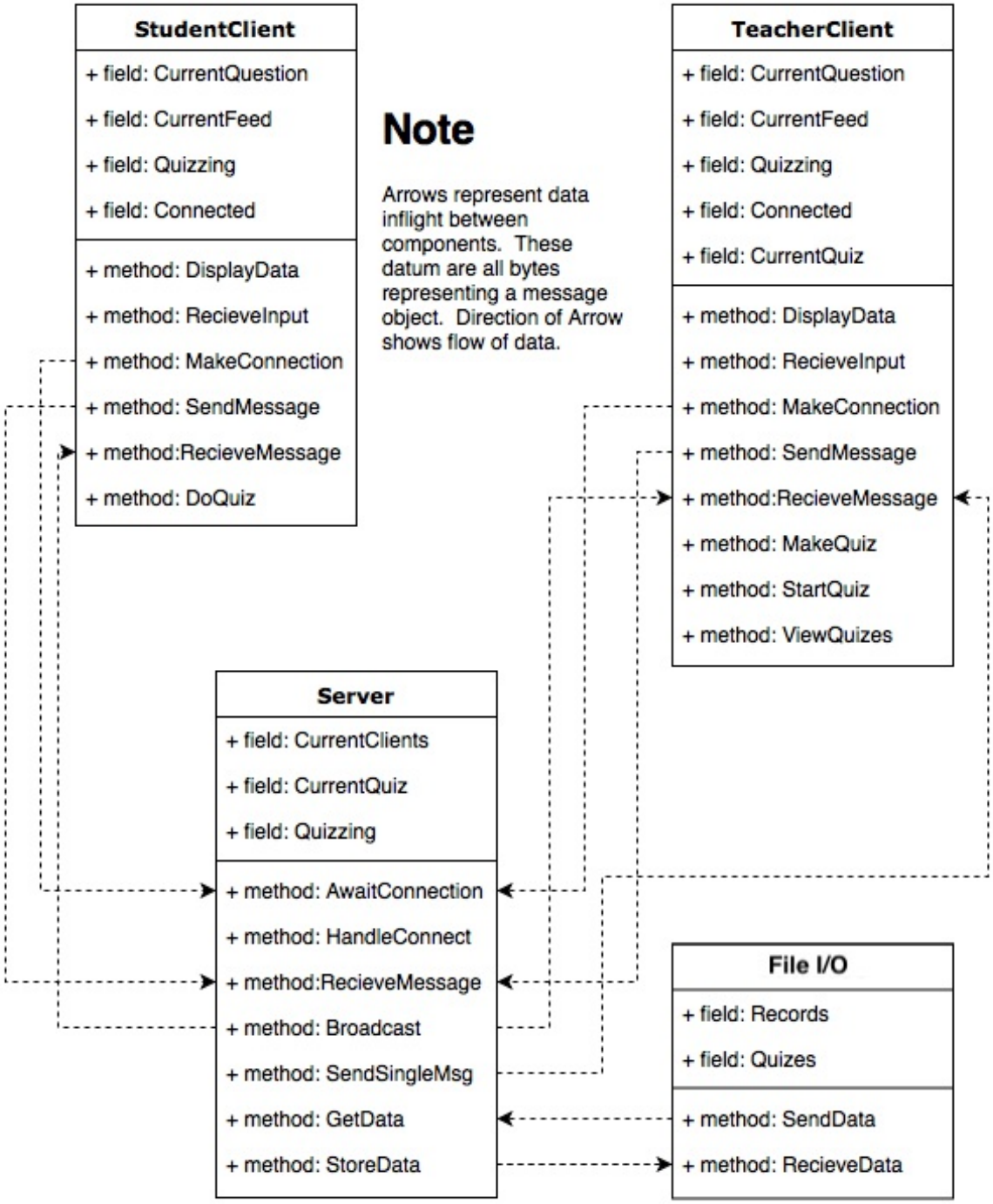
communications, and for displaying communications. Here, other methods will be provided for parsing communications and for forming messages for our protocol. The user interface will have data for the current state of the program, such as current quizzes, current quiz questions, current chat feed, and how many students are present.

Server

The Server will possess an up to date log of current connection addresses. The server will have a method to await connections that is always running during operation. This method will spawn threads of processing that have different methods to handle a new connection, to send messages to individuals, to send messages to all clients, to recieve messages from individuals, and to perfom quizes. The server will have methods for communicating with the database in order to store and retrieve quizes as well as statistics related to those quizes.

File I/O

The File I/O will hold premade quizes as well as class performance on said quizes. The file system will have methods for storing and retrieving this data for the network to pass along to the user interface.



Quality Assurance Plan

1. Reviews

    i. The manager of the team will review project plan, identify if it is realistic, and evaluate readability.
    ii. Each member of the team will review SDS and SRS whether the design capture all the requirements and identify areas that need to be improved
    iii. Each member of the team will review source code for the project to identify if all SRS

documents are implemented

iv. Each programmer will perform unit test for each modules during developing as well as GUI for errors
v. Technical Writer will review software documentation(User Manual) and evaluate if it is easy to understand for users

2. Testing

To ensure our software meets SRS, testing will be done in the following list:

1. Test Process

    i. Login/Logout
    ii. Posting/Receiving questions
    iii. Publishing quiz
    iv. Submitting quiz
    v. Storing quiz

2. Outline of Test Cases

    i. Login/Logout

        a. Verify user can log in
            a. Steps:
            b. Enter name and identification to log in
            c. Expected Result:
            d. User can start out on main chat page
        b. Verify user can log out
            a. Steps:
            b. Click log out button
            c. Expected Result:
            d. User is exited from the application
        c. Verify user cannot start out on chat page without login information being typed
            a. Steps:
            b. Click login button without identification or IP address
            c. Expected Result:
            d. User is not able to access application

    ii. Sending messages to teacher

        a. Verify user(students) can send messages to teacher
            a. Steps:
            b. Log in with user name and identification

c. Type question into input box
d. Click send button
e. Expected Result:
f. Professor receives the question

b. Verify user(teacher) can receive questions
    a. Steps:
    b. Log in with name
    c. Expected Result:
    d. User can see list of the messages that are sent by users(Students)

iii. Publishing quiz

a. Verify user(Teacher) can publish a quiz
    a. Steps:
    b. Log in with name
    c. Create quiz questions
    d. Create multiple choice quiz
    e. Click "Send"
    f. Expected Result:
    g. User(Student) receives the quiz

iv. Submitting quiz

a. Verify user(Students) can open and submit a quiz
    a. Steps:
    b. Log in with user name and IP Address
    c. Click one quiz among list of quizzes
    d. Answer the quiz
    e. Click "Submit" button
    f. Expected Result:
    g. The quiz is erased off of the student view

v. Storing quiz

a. Verify all created quizzes are stored in the file system
    a. Steps:
    b. Log in with name
    c. User(Teacher) creates a quiz
    d. User logs out and logs back in
    e. Expected Result:
    f. User(Teacher&Students) can see all the list of premade quizzes