**mASK Requirements**

**Introduction**

Background Information

See domain analysis for background information

Environment and System Models

The platform for this application will be on both mobile devices and PC/Mac OS. The front end will be created using the NetBeans IDE (utilizing the Java programming language) and the back-end will be a combination of using Microsoft SQL Server Management Studio 17 and AWS Cloud Computing/Storage. It will be created using Microsoft OS.

**Functional Requirements**

1. The front-end of the application will allow users to view questions asked by classmates, ask questions, rank questions based upon a voting scheme, and answer questions
2. The front-end of the application will allow the host (professors) to ask questions, manually manipulate the ranking of questions, and answer questions
3. Both the user and host (student and professor) will have access to data manipulated by the Data Storage/Manipulation subsystem
4. The back-end allows for the storage, retrieval, and manipulation of the data received from the front-end application
5. The back-end will perform data analytics to provide feedback for individual users as well as system administrators (i.e. average usage, voting trends, etc.)
6. Future iterations of the application will include machine learning techniques to analyze trends to make predictions and classifications (regarding users)

**Quality Requirements**

1. The front-end of the application will be designed for swift and efficient use, as well as ease of use regarding user control
2. The database will be designed efficiently to provide swift retrieval and storage of the questions, answers, and personal information.
3. In case of data corruption/loss there will always be at least one back-up server available with constant updates
4. The personal information of the users will be securely stored and transferred using AWS, along with strict access requirements to prevent privacy issues

**Platform Requirements**

1. Computing Platform: As a Java-based platform, the application should be able to run on any computer system
   1. The specific computing technologies used include: JavaFX with CSS, an IDE such as NetBeans
2. Server: The original server will be created using Microsoft SQL Server Management. This is an IDE (Integrated Development Environment) that implements relational databases and servers using SQL.
3. Cloud Computing: Amazon EC2 cloud computing will be used to allow multiple network access to the server created for the application. This is a cloud server provided through AWS.
4. Storage: The server will need a constantly updated back-up server in case a primary server crash occurs, either wiping or corrupting the primary data.
5. Hardware: The user will be required to use a keyboard/mouse or touch-input computer to navigate the application

**Process Requirements**

The project must be completed by April 28th, 2019 with a specification document due March 12th and source code due March 28th.

**Domain Analysis**

Introduction

This application’s purpose is to provide a decentralized query process for students in a lecture-type environment. The students ask a question via a mobile/web-based application, answer other students’ questions, then rank the questions based upon voting rates. The professor can access questions, manually rank the questions, and answer questions as deemed fit. A back-end portion of this application exists to provide data storage along with data manipulation.

Glossary

**AWS (Amazon Web Service):** A cloud services platform that allows users access to computing power, cloud storage, content delivery, and other various functionalities

**Client:** The users connected to the host device. The clients can ask questions, as well as answer questions

**Cloud Computing/Storage:** On-demand storage of data and computing power via the internet; information is stored in a server (‘cloud”) by the provider (in this case AWS)

**CSS:** A tool used to add visual components to applications

**Data Analytics:** The analyzation of information given using various statistical measures (i.e. mean, standard deviation, general trends, etc.)

**Database:** an electronic storage system that houses the data provided by the application

**Host:** The person who starts the session. They can push questions to all client devices as well as close questions asked by the clients

**IDE:** An integrated development environment. This is the software that is used to create a database and develop its behaviors

**NetBeans:** An IDE used for Java development

**Server:** This is a centralized point for a network to where data can be stored and retrieved. Databases are typically stored in servers.

**System Administrator:** The operator of the back-end part of the application. The individual(s) takes care of the upkeep and design of the overall subsystem.

Extensions

Further extensions of this application would include point systems, incentives to ask questions, and machine-learning analytics to automatically rank questions based upon subject/need.

General Knowledge about the Domain

1. Host will usually be a professor during a lecture
2. The clients will usually be students in the lecture
3. Setting will be in a lecture-based environment
4. Data included will be student ID, IP address, host name, users tied to host name, cloud network address, server name

Client and Users

1. System administrators can use data to predict trends and provide data for the students and professors
2. The users of this application will include:
   1. Professors, or lecturers
   2. Students
   3. Businesses, training workshops

The Environment

The environment for this subsystem will be across the Internet (via cloud storage/computing). The initial program will be developed in the Java programming language, capable of being used across all platforms that support this language.

Tasks and Procedures

The differing usages of the application are as follows:

1. The student (client) will ask questions, answer questions, vote on questions, and have access to their personal data
2. The professor (host) will ask questions, answer questions, manually manipulate questions, and have access to student data
3. The system administrator will have access to the database/server and be able to update the back-end framework of the application.

Competing Software

There are multiple applications in existence that are similar to the design, such as iClicker and Kahoot!. This software will have qualities very similar to these technologies (i.e. server storage for data, decentralized usage) but hopes to create an environment to ask questions instead of being quizzed.

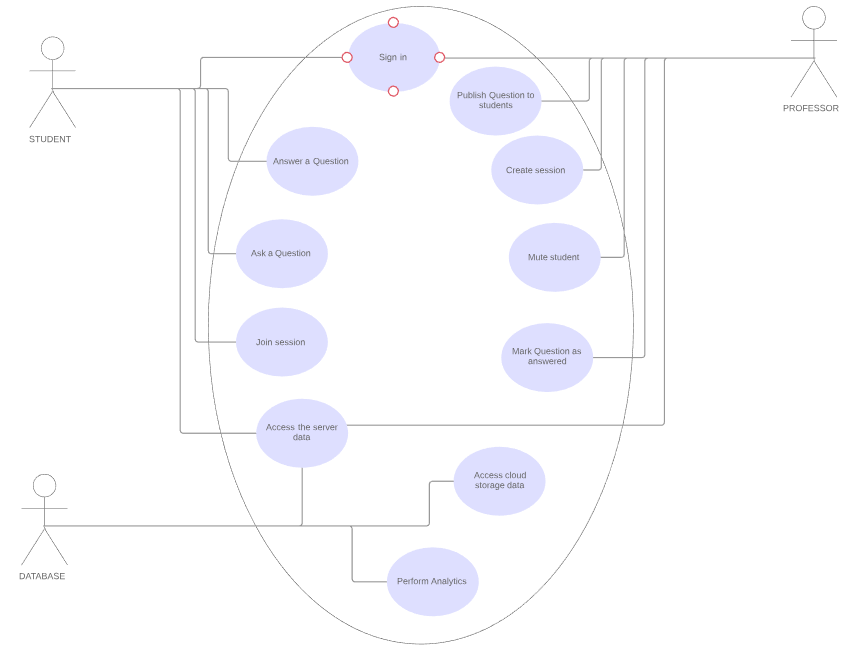
Domain Similarities

There are many similar applications such as iClicker and Kahoot!. The storage of data received will be very similar (however data analytics might differ), the usage of mobile devices and computers is the same, and a similar hierarchy between host and user will exist. However, this application hopes to decentralize the query process of students that does not exist within these similar technologies.

**Use Cases**

1. Case: Sign-in
   1. Actors: Student, Professor
   2. Preconditions: App must be loaded on user’s device and on the home screen
   3. Related use case: None
   4. Steps:
      1. Provide username and password
      2. Press enter for verification
2. Case: Answer a Question
   1. Actors: Student
   2. Preconditions: Must be on a question’s page where the “answer question” button is
   3. Related use case: Ask a question
   4. Steps:
      1. Type out answer for a question in text box provided
      2. Press submit
3. Case: Ask a Question
   1. Actors: Student
   2. Preconditions: Be on the event screen that prompts user to ask a question
   3. Related use case: Answer a Question
   4. Steps:
      1. Type out question in text box provided
      2. Press submit
4. Case: Join Session
   1. Actors: Student
   2. Preconditions: Must be in a valid environment for server presence and on the “join session” screen
   3. Related use case: Create Session
   4. Steps:
      1. Get session information and provide it in the application
      2. Press submit and session should be joined
5. Case: Access Server Data
   1. Actors: System admin
   2. Preconditions: Server must have received data from front-end application
   3. Related use case: Access Cloud Data
   4. Steps:
      1. Login to system
      2. Query for a specific data set
6. Case: Access Cloud Storage Data
   1. Actors: System admin
   2. Preconditions: Cloud storage system must have received data from the server and analytics
   3. Related use case: Access Server Data
   4. Steps:
      1. Login to cloud system
      2. Query for a specific data set
7. Case: Perform Analytics
   1. Actors: System admin
   2. Preconditions: Data must have been received from the cloud storage unit
   3. Related use case: none
   4. Steps:
      1. Model Data
      2. Further analyze using classification algorithms
8. Case: Publish Question to Students
   1. Actors: Professor
   2. Preconditions: Be on publish question page that prompts user to do so
   3. Related use case: Ask a Question
   4. Steps:
      1. Type out text of question to be published
      2. Press submit
9. Case: Create Session
   1. Actors: Professor
   2. Preconditions: Be on home page where “create session” button is prompted
   3. Related use case: Join Session
   4. Steps:
      1. Provide application with proper session information
      2. Press submit
10. Case: Mute Student
    1. Actors: Professor
    2. Preconditions: Be on the question/answers page where student’s responses are displayed
    3. Related use case: Mark Question as Answered
    4. Steps:
       1. Toggle button next to student’s username to mute student
11. Case: Mark Question as Answered
    1. Actors: Professor
    2. Preconditions: Be on question/answers page
    3. Related use case: Mute Student
    4. Steps:
       1. Toggle button next to question header to mark question as answered

**Use Case Diagram**

****

**SUBSYSTEM: Data Storage/Manipulation** (Alexander Darden)

**Introduction**

Background Information

See domain analysis in appendix for background information

Environment and System Models

The storage platform for this application will be implemented using Microsoft SQL Server Management Studio 17. A server will be required to access the information from multiple networks, which will also require the application itself (using the Java programming language) access to this server. The server will be created on a Microsoft OS and use Amazon EC2 cloud computing to allow multiple network access.

**Functional Requirements**

1. The system allows for the storage, retrieval, and manipulation of the data received from the front-end application
2. The server side will perform data analytics to provide feedback for individual users as well as system administrators (i.e. average usage, voting trends, etc.)
3. Future iterations of the application will include machine learning techniques to analyze trends to make predictions and classifications (regarding users)

**Quality Requirements**

1. The database will be designed efficiently to provide swift retrieval and storage of the questions, answers, and personal information.
2. In case of data corruption/loss there will always be at least one back-up server available.
3. The server will periodically be checked and serviced to prevent oversaturation of unnecessary data.
4. As the application is updated, features to the server will be correspondingly updated as well

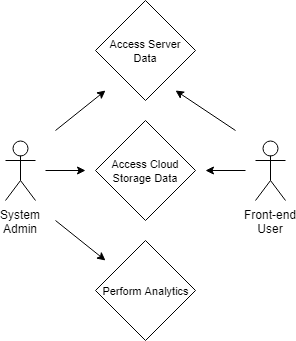
**Platform Requirements**

1. Server: The original server will be created using Microsoft SQL Server Management. This is an IDE (Integrated Development Environment) that implements relational databases and servers using SQL.
2. Cloud Computing: Amazon EC2 cloud computing will be used to allow multiple network access to the server created for the application. This is a cloud server provided through AWS.
3. Storage: The server will need a constantly updated back-up server in case a primary server crash occurs, either wiping or corrupting the primary data.

**Process Requirements**

The process requirements for this subsystem are the same as the main document. The due dates do not differ between subsystems for this assignment.

**Use Case Diagram**



**Use-Cases**

1. Use case: Access Server Data
   1. Actors: System admin, Front-end user
   2. Preconditions: Server must have received data from front-end application
   3. Related use case: Access Cloud Data
   4. Steps:
      1. Login to system
      2. Query for a specific data set
2. Use Case: Access Cloud Data
   1. Actors: System admin, Front-end user
   2. Preconditions: Cloud storage system must have received data from the server and analytics
   3. Related use case: Access Server Data
   4. Steps:
      1. Login to cloud system
      2. Query for a specific data set
3. Use Case: Perform Analytics
   1. Actors: System admin
   2. Preconditions: Data must have been received from the cloud storage unit
   3. Related use case: none
   4. Steps:
      1. Model Data
      2. Further analyze using classification algorithms

**Domain Analysis**

Introduction

This subsystem provides an overview of the data storage, data retrieval, and data manipulation processes. These are the back-end processes used by the main application.

Glossary

**AWS (Amazon Web Service):** A cloud services platform that allows users access to computing power, cloud storage, content delivery, and other various functionalities

**Cloud Computing/Storage:** On-demand storage of data and computing power via the internet; information is stored in a server (‘cloud”) by the provider (in this case AWS)

**Data Analytics:** The analyzation of information given using various statistical measures (i.e. mean, standard deviation, general trends, etc.)

**Database:** an electronic storage system that houses the data provided by the application

**IDE:** An integrated development environment. This is the software that is used to create a database and develop its behaviors

**Server:** This is a centralized point for a network to where data can be stored and retrieved. Databases are typically stored in servers.

**System Administrator:** The operator of the back-end part of the application. The individual(s) takes care of the upkeep and design of the overall subsystem.

Extensions

This subsystem can be used for further adaptations of the overall application. If there is a future update that implements functionalities not currently included, the database system should be able to account for data gathered.

General Knowledge about the Domain

1. Data included will be student ID, IP address, host name, users tied to host name, cloud network address, server name
2. The data will be of the String type for the most part, with certain int values (i.e. student ID)

Client and Users

1. System administrators can use this data to predict trends and provide data for the students and professors
2. Students can use this data to measure their activity and what topics seem to require the most questions
3. Professors can use this data to measure activity level of students as well as complex topics that need review

The Environment

The environment for this subsystem will be across the Internet (via cloud storage/computing). Our original server will be created via Microsoft SQL Server Management Studio on a Microsoft OS but will be transferred to the cloud system using AWS. The system administrator will need access to a computerized device in order to access this information.

Tasks and Procedures

The storage, retrieval, and manipulation of data is as follows:

1. The storage of data will come from the front-end of the application. This will include:
   1. Questions being asked from both the professor and student side of the application
   2. Answers from the student and professor side of the application
   3. Voting from the student side of the application
   4. Question manipulation from the professor side of the application
   5. Personal data from the student side of the application
   6. Data from the host side (server name, address)
2. The storage of the data will be housed in a database (created through Microsoft SQL Server Management Studio) that is then sent to cloud storage via AWS
   1. Analytics will be performed here and consequentially stored in the server as well
   2. System Administrators will have access to this data and decide whether it is negligible or can be sent back to the front-end of the application
3. The data is retrieved from the server and returned to the front-end of the application
   1. The student shall receive data about their usage, questions asked, etc.
   2. Professors will get data for each student regarding activity, along with data about what particular dates seemed to have the most questions asked (along with further updates)

Competing Software

There are multiple database technologies, as well as cloud-based technologies in the market today, but this subsystem implements these technologies as opposed to creating something new. For instance, we use AWS as our cloud storage and use Microsoft SQL Server Management Studio as our IDE.

Domain Similarities

There are other applications that use the same domain (students, professors, etc.) that are in a similar environment to this application’s (Poll Everywhere, iClicker, etc.). The data systems for these applications more than likely would be very similar to this application’s, but the point of our application is not to create an innovative data storage system, but rather decentralize the query process of questions being asked in class.

**SUBSYSTEM: THE PROFESSOR ASKS A QUESTION TO THE STUDENTS (MCKENZIE MOIZE)**

**Functional Requirements**

1. The system allows for the professor to ask a question that will be pushed to student devices, allowing them to answer with multiple choice or short answer.
2. The user will use a text field to enter a String containing the question
3. The user will see a button field where they can decide to have the students answer as short answer or multiple choice
   1. If multiple choice is selected, a set of text fields will appear where the answer choices are shown
4. The user will use a text field to input the time limit the students have to answer the question
5. The user will use a button to push the question to the student devices

**Quality Requirements**

1. In the event of internet loss
   1. The class analytics will saved onto the server-side and re-uploaded whenever a connection is established
   2. The professor will be kicked back to the join/create group screen, with an option to restart with the old session ID.
2. The system will not allow the professor to post the question without provided answer choices if multiple choice is selected.
3. The system will prompt to be checked for updates after a session ends.
4. The system will use minimal resources so that it does not slow down the instructional material

**Platform Requirements**

1. Computing Platform: As a Java-based platform, the application should be able to run on any computer system. Later updates will add cross-platform mobile support.
2. Technology Used: The graphical interface will be created using JavaFX with CSS components added for visual effects. The algorithms and coding will be completed in Java using an IDE such as NetBeans. The backend database will be pipelined to MS-SQL. We will GitHub for version control and pushing updates to users.
3. Hardware Requirements: The user will be required to have a keyboard/mouse or touch-input computer so that they can navigate the application.

**Process Requirements**

The process requirements for this subsystem are the same as the main document. The due dates do not differ between subsystems for this assignment.

**Appendix**

**Introduction**

This subsystem provides an overview of the interface used by the host of the current session to ask questions of the clients.

**Glossary**

Client: The users connected to the host device. They are able to ask questions of the professor as well as answer the questions he asks.

CSS: A tool used to add visual components to applications.

GitHub: A version-control software. It is used to upload a program and its components to a centralized internet location.

Host: The person who starts the session and is in charge of it. They can push questions to all client devices as well as close client-asked questions.

IDE: An integrated development environment. It is used to more efficiently develop programs.

NetBeans: An IDE used for Java development.

Session: The current group of clients and host. Each group will have their own session ID and will be able to use it to connect with each other.

Server: The centralized location of all session information.

**Extensions**

Extensions of this subsystem would be implementing a point system for speed in which the questions were answered and other ways to gauge client interaction.

**General Knowledge About the Domain**

1. The host will usually be a professor during a lecture.
2. The clients will usually be students in the lecture.
3. The setting of the software will primarily be a classroom/lecture hall

**Clients and Users**

1. Potential users of this software would be classrooms with a large amount of students or professors who wish to automate the process of analyzing student participation throughout the semester.
2. The users of this subsystem will be the hosts. This can include
   1. Professors in a class
   2. Students in a student-led tutoring session
   3. Business heads during a meeting where participation is required, such as polling.

**The Environment**

1. The application will be broadcasted over the internet, therefore each user will need to be connected.
2. The initial program will be developed in Java, which can be used on all computing platforms that contain a JVM.

**Tasks and Procedures**

1. The question will come from the host
2. The host will input the type of question: short answer or multiple choice
   1. If multiple choice the host will be prompted to enter the answer choices
3. The host will enter the time limit for the question, if there is one
4. The host will press a button to push the question to the students devices

**Competing Software**

There are multiple applications in existence that are similar to the design, such as iClicker and Kahoot!. The subsystem will be modeled similar to these applications, but seeks to take the best qualities of each software and centralized it into one application.

**Domain Similarities**

There are many similar applications such as iClicker or Kahoot!, but the point of our application is to have a centralized experience for the classroom. There will be one app to do all the functions, as opposed to having multiple platforms used during a single class session.

**Subsystem - Students Can Submit Questions (Emily Ashburn)**

**Introduction**

Background Information

See Domain Analysis in appendix for background information.

**Functional Requirements**

1. The system allows Students to ask questions related to class discussion or class lecture.
   1. The system allows Students to create a title and short description for a new question.
   2. When Students create a question, the system forces a 75 character limit on the question’s title.
   3. The system allows Students to create a description that defines his/her question in-depth without a character limit.
   4. The system does not allow Students to delete their question after submitting it to the Forum.
   5. The system will enforce a vulgar filter upon question creation.
      1. The vulgar filter will asterisk out the inappropriate word after the question is submitted.
      2. The vulgar filter will not kick out the Student from the session. There are no repercussions to submitting inappropriate questions.
2. The system allows Students to respond to each other’s questions
   1. The system does not force a character limit on the Student’s reply.
   2. The system enforces a vulgar filter upon replies.
   3. This permission can be toggled by the Teacher.

**Quality Requirements**

1. In the case of failure of internet connection during the creation of a question:
   1. The system will discard the Student’s question completely.
   2. The questions submitted by the Student previously will remain in the forums and will not be deleted upon disconnection.
   3. The Student will be disconnected from the session and will need to re-enter with the session group ID generated by the Teacher.

**Platform Requirements**

1. Computing Platform: The application will be able to run on any machine capable of supporting a web-based browser such as Chrome, Firefox, Safari, and Internet Explorer/Edge.
2. Technology Used: This system’s algorithms and code will be written using Java within the IDE NetBeans. This system’s GUI will be written using JavaFX with CSS.
3. Hardware Requirements: The user will need a computer with a keyboard/mouse or touchscreen computer to use the application.

**Process Requirements**

The process requirements for this subsystem are the same as the main document. The due dates do not differ between subsystems for this assignment.

**Domain Analysis**

Introduction

This document describes and explains one subsystem created for the non-host Student user in detail. This system allows Student users to create and submit questions, directed to the host Teacher user, to the public forums of the current session.

Glossary

**Browser:** A software application for accessing information on the World Wide Web.

**CSS:** A tool used to add visual components to applications.

**Forums:** The session’s bulletin board where all questions, created by Teacher or Students, will appear.

**GUI:** A Graphical User Interface. A form of a user interface that allows users to interact with graphical icons and visual indicators.

**Host:***See* ***Teacher***

**IDE:** An Integrated Development Environment. This is used to efficiently develop programs.

**Java:** An object-oriented programming language

**JavaFX:** A software platform for creating desktop applications.

**NetBeans:** An IDE use for Java development.

**Session:** A term that refers to the group of Students and Teacher using this application. Each group has a unique group ID generated by the Teacher and other Students can connect only with the unique group ID.

**Student:** A user that participates in the session. This user’s identity is anonymous to other students. This user can also create questions for the Teacher to answer, upvote other Students’ questions, and reply to other Students’ questions.

**Teacher:** A user that leads the session. This user is normally the teacher/professor/lecturer in a classroom setting.

**Use-case:** A term to describe the different possible users that will use this application (Student, Teacher, System Admin, etc)

**User:** A general term to refer to either the Student or Teacher/Host use-case.

**Vulgar filter**: A custom algorithm to censor inappropriate words determined by the developers.

Extensions

Possible extensions of this application will include cross-platform functionality to run on Android and Apple smartphones.

General Knowledge about the Domain

1. The host of a session is normally the teacher/professor/lecturer
2. This system works for classrooms with small or large amounts of students
3. Teachers can analyze the statistics generated by the Students to determine participation, growth, and attendance.
4. Students are anonymous to other students and have no unique or defining trait to his/her question to reveal his/or identity.

Client and Users

1. Students have permissions to
   1. Create questions
   2. Upvote existing question
   3. Reply to questions
2. Teachers have permissions to
   1. Create questions
   2. Delete questions
   3. End session
3. System Administrator have permissions to
   1. View data submitted by Students and Teachers

The Environment

1. The system will be accessible over the internet, to which each user needs to be connected.
2. The main program will be written in Java, which can be handled on all platforms that have a JVM installed. (Linux, Mac OS, Solaris, Windows platforms, etc.)

Tasks and Procedures

1. Student clicks a “Create Question” button
2. Students will enter a short title (75 characters) and description explaining their question in more detail.
3. Student submits a question to the Forum where other students and the teacher can view.
4. Once the Student submits their question, they cannot retract or delete the question from the forum.

Competing Software

There are a few applications that have similar aspects to this application, such as Kahoot!, iClicker, and Reddit. The innovation of this application mixes the best components of these applications along with new ideas to provide a safe environment for students to ask any and every question they have pertaining to the lecture.

Domain Similarities

There are many similarities between Kahoot!, iClicker, Reddit, and this application. This system uses the anonymous aspect from Kahoot!, polling aspect from Kahoot! and iClicker, and forum and upvoting aspect from Reddit.