FALL 2014 GROUP 1

SOFTWARE DESIGN DOCUMENT

Twitter Project

Patrick Burton

Arvind Nair

Lakshmi Swathi Chavvakul

CSCI 50600

IUPUI

1 Table of Contents

2 Introduction

2.1 Purpose

This Software Design Document describes in detail the architecture of the software system and also the Use Case and sequence diagrams of each of the four iterations which we have done as we had adopted an agile development process. This document is intended for a clear understanding of our software system and its working for the users and also for future extensions as and when needed.

2.2 Scope

The software which we developed is a simple messaging application which will help users to perform the following basic activities:

1. The user can after logging in post tweets or messages.

2. He can read messages posted by others also.

3. He can perform these functions from Desktop, PC and mobile devices.

2.3 Methodology

For developing the software, we have followed an agile methodology of 4 iterations. For each of the iterations we have implemented the basic requirements but added onto them the extra features depending on the requirements and metric measurements to improve the quality of software. We have gradually added and improved the features and it is shown in the metrics measurements documents.

2.4 Abbreviations

1. CAS : Central Authentication Server

2. IUPUI: Indiana University Purdue University-Indianapolis

3. MAC: Math Assistance Center

4. AWS: Amazon Web Services

3 Detailed Software Design Description in Iteration 1

3.1 Use Case Model (General View)

3.1.1 Diagram

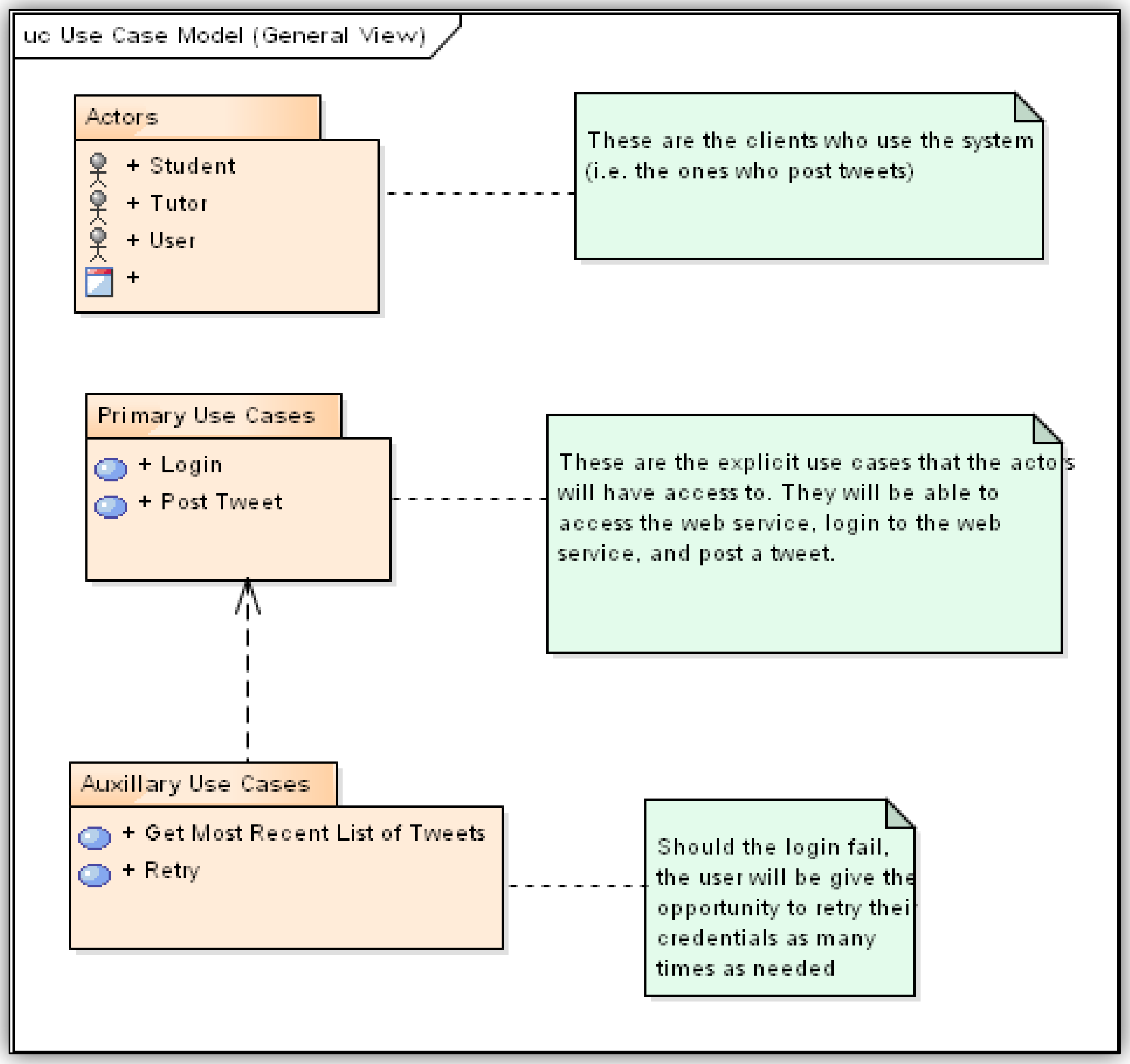

uc Use Case Model (General View)
Actors
+ Student
+ Tutor
+ User
+
These are the clients who use the system (i.e. the ones who post tweets)
Primary Use Cases
+ Login
+ Post Tweet
These are the explicit use cases that the actors will have access to. They will be able to access the web service, login to the web service, and post a tweet.
Auxillary Use Cases
+ Get Most Recent List of Tweets
+ Retry
Should the login fail, the user will be give the opportunity to retry thei credentials as many times as needed

Figure 1 : Use Case Model (General View) - Iteration 1

3.1.2 Description

Actors: The actors are the users who will use the system. They are student, tutor and user.

Primary Use Cases: It shows the functions of login and posting tweet by any of the actors.

Auxillary Use Cases: It shows additional scenarios or functions like getting most recent tweets and retrying if the login fails.

3.2 Actors View

3.2.1 Diagram

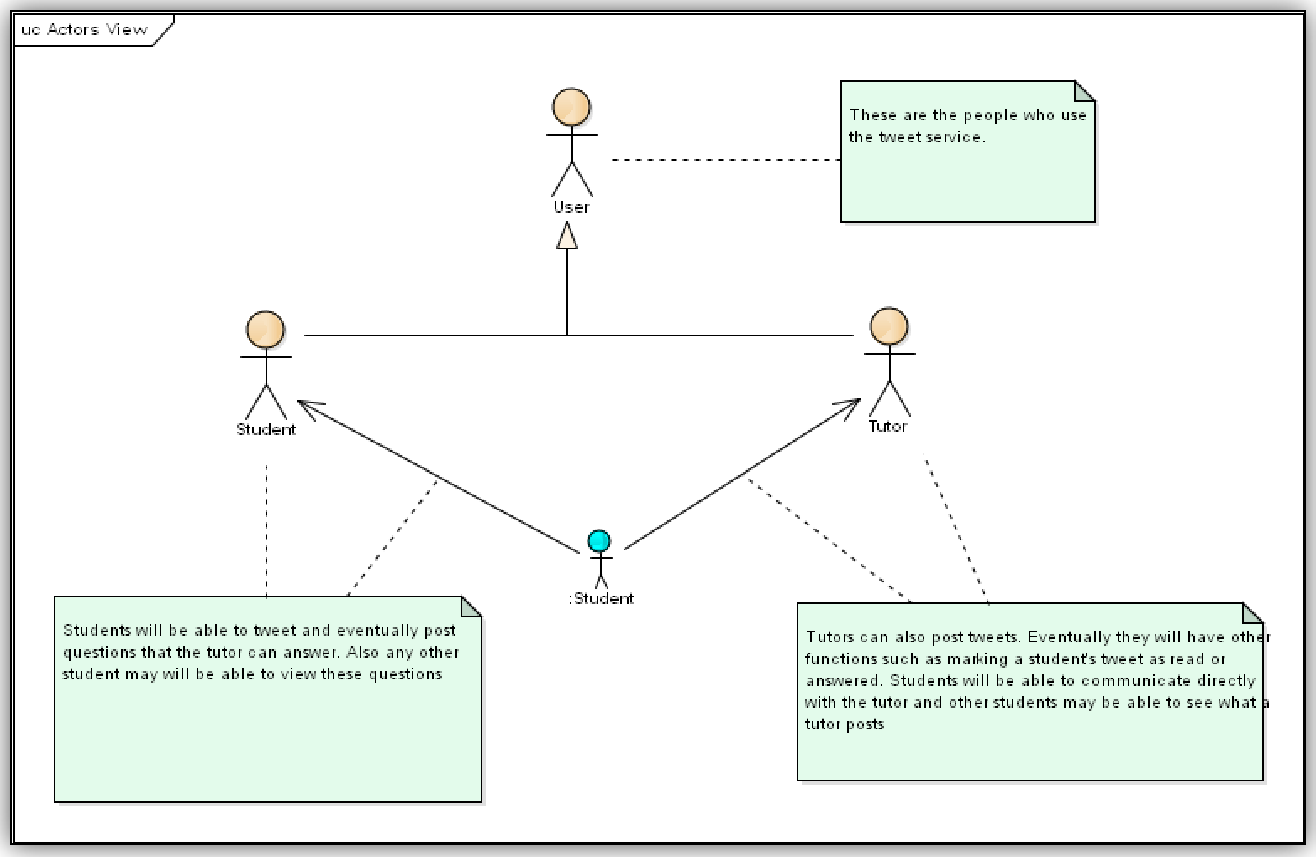

uc Actors View
User
These are the people who use the tweet service.
Student
Tutor
:Student
Students will be able to tweet and eventually post questions that the tutor can answer. Also any other student may will be able to view these questions
Tutors can also post tweets. Eventually they will have other functions such as marking a student's tweet as read or answered. Students will be able to communicate directly with the tutor and other students may be able to see what a tutor posts

Figure 2 : Actors View - Iteration 1

3.2.2 Description

It shows the in detail the user having types of student and tutor.

3.3 Primary Use Case View

3.3.1 Diagram

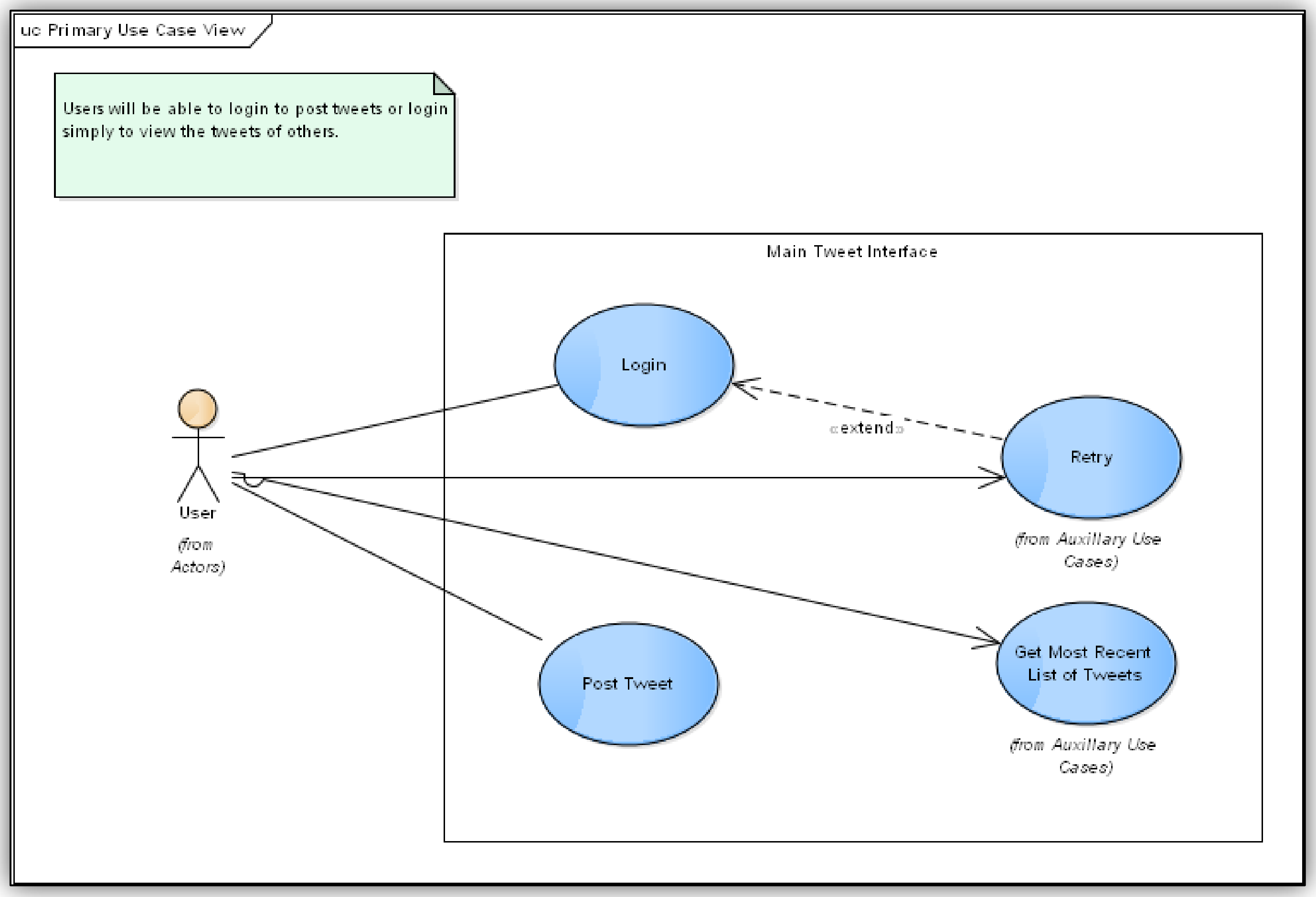

uc Primary Use Case View
Users will be able to login to post tweets or login simply to view the tweets of others.
Main Tweet Interface
Login
«extend»
Retry
(from Auxiliary Use Cases)
User
(from Actors)
Post Tweet
Get Most Recent List of Tweets
(from Auxiliary Use Cases)

Figure 3 : Primary Use Case View - Iteration 1

3.3.2 Description

Here, the functions which can be performed by the user like login, posting tweet, retrying and getting most recent tweets are shown.

3.4 Retry View

3.4.1 Diagram

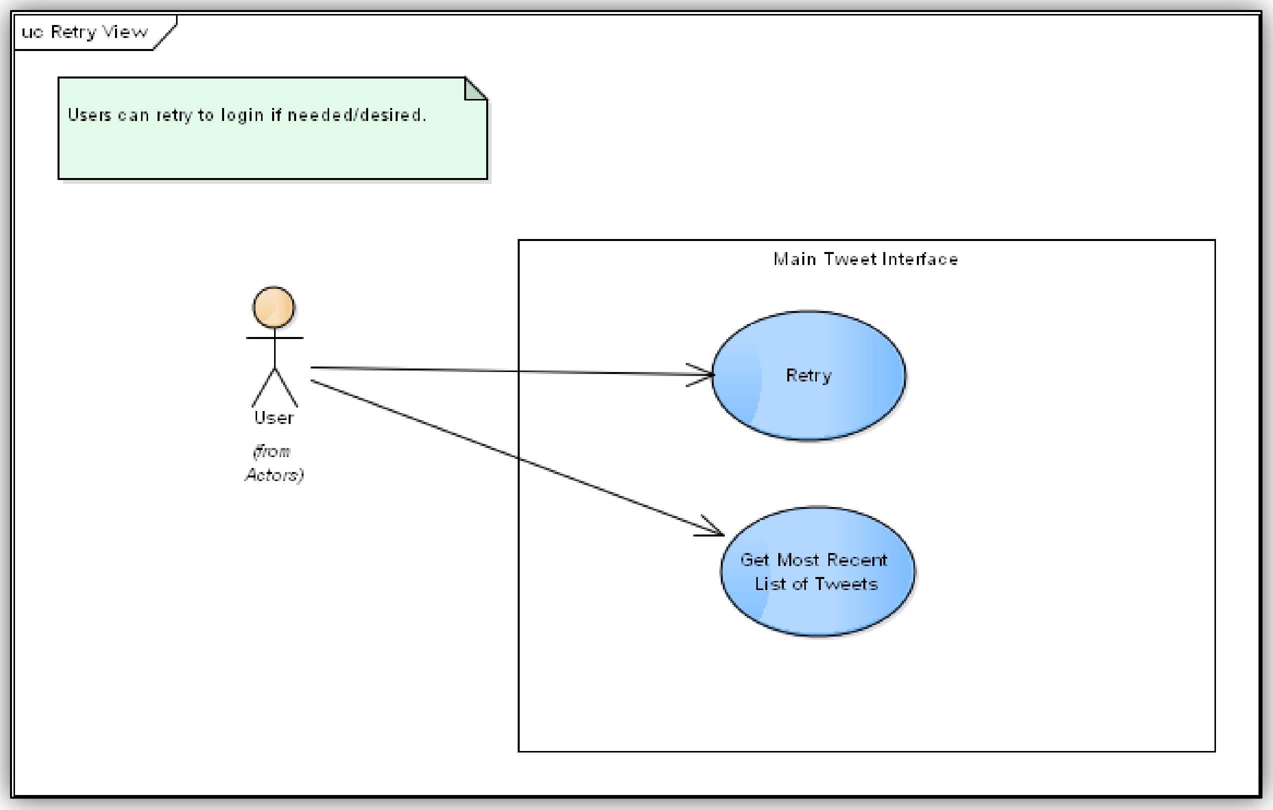

uc Retry View
Users can retry to login if needed/desired.
Main Tweet Interface
User
(from Actors)
Retry
Get Most Recent List of Tweets

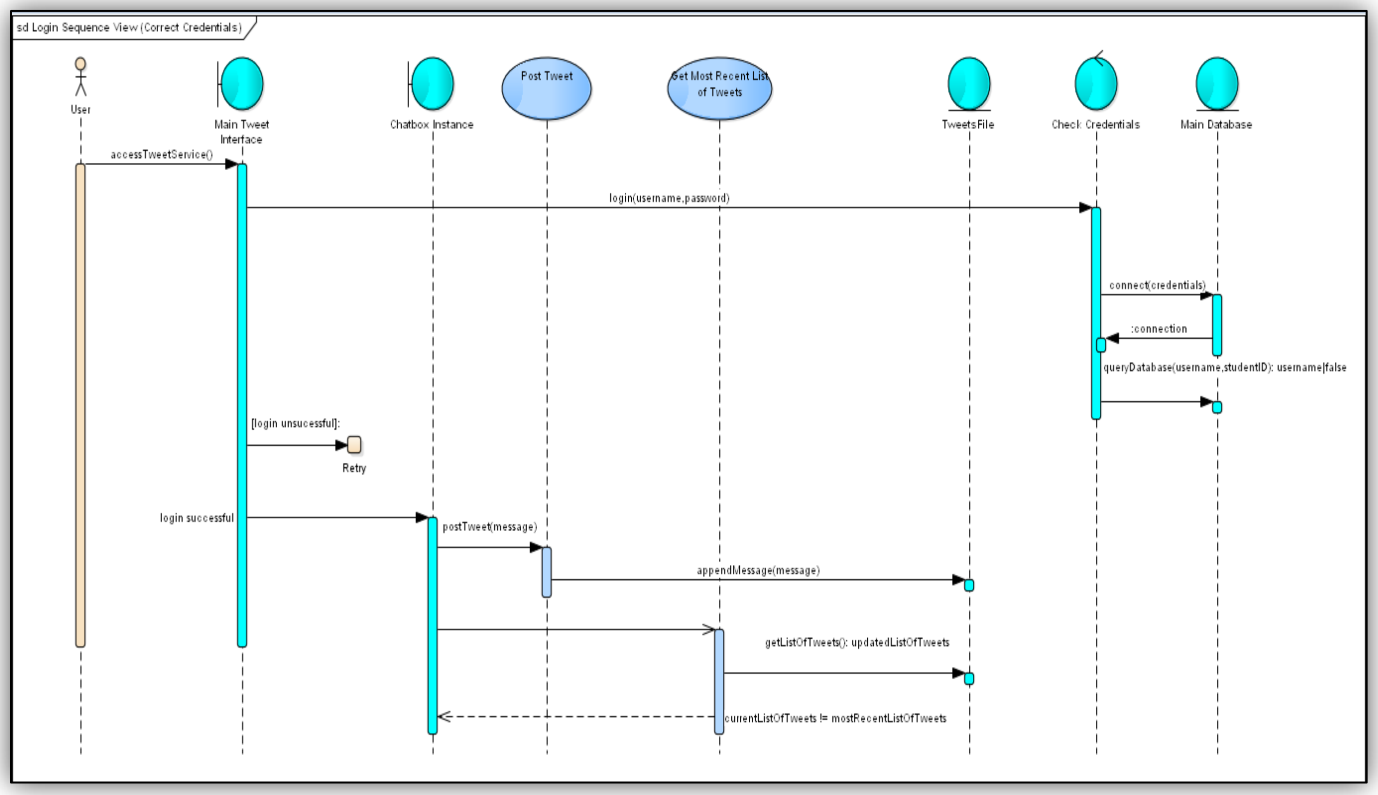
Figure 4 : Retry View - Iteration 1

3.4.2 Description

Here, the user can retry and once successful, can get the most recent tweets.

3.5 Login Sequence View (Correct Credentials)

3.5.1 Diagram



The user attempts to log in, this login will be checked against a main database and the user wil be able to retry. If eventually successful, they will be given an instance of a chat box to post tweets.

Figure 5 : Login Sequence View (Correct Credentials) - Iteration 1

3.5.2 Description

The sequence diagram above shows the events in the system. They are described below:

1.The user first accesses the website by visiting the URL.

2. He enters his credentials in the username and password field and clicks the login button.

3. The credentials are checked with the database and then once approved the user is taken to the main chat interface.

4. Here, he can type in the message box and press enter and the message is tweeted out and displayed on the main message wall.

5. The tweets are stored in a tweets file and the user can get the most recent tweets from them.

3.6 Login Sequence View (Retry Sequence)

3.6.1 Diagram

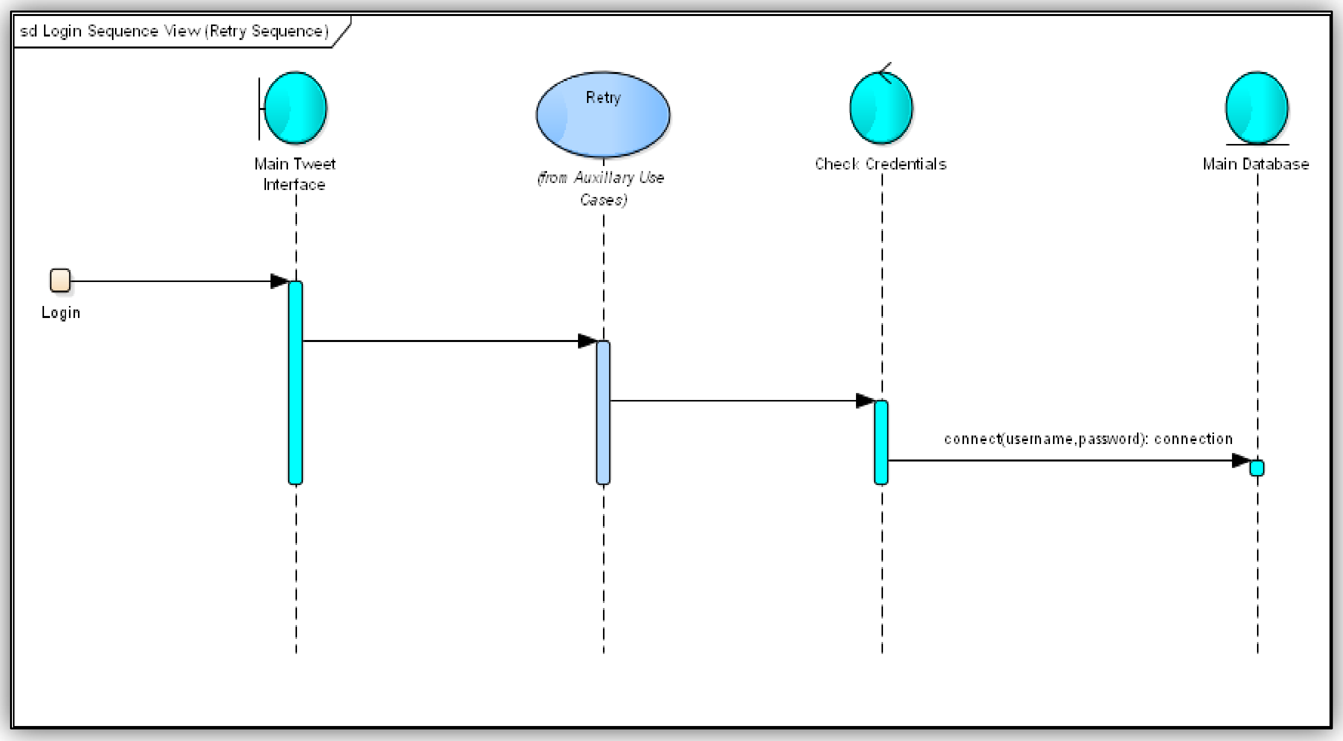

sd Login Sequence View (Retry Sequence)
Main Tweet Interface
Retry
(from Auxiliary Use Cases)
Check Credentials
Main Database
Login
connect(username,password): connection

Figure 6 : Login Sequence View (Retry Sequence) - Iteration 1

3.6.2 Description

This just shows the retry scenario in which the user has to retry and the credentials are checked with the database. If they match, the user can login otherwise he has to retry again.

4 Detailed Software Design Description in Iteration 2

4.1 Use Case Model (General View)

4.1.1 Diagram

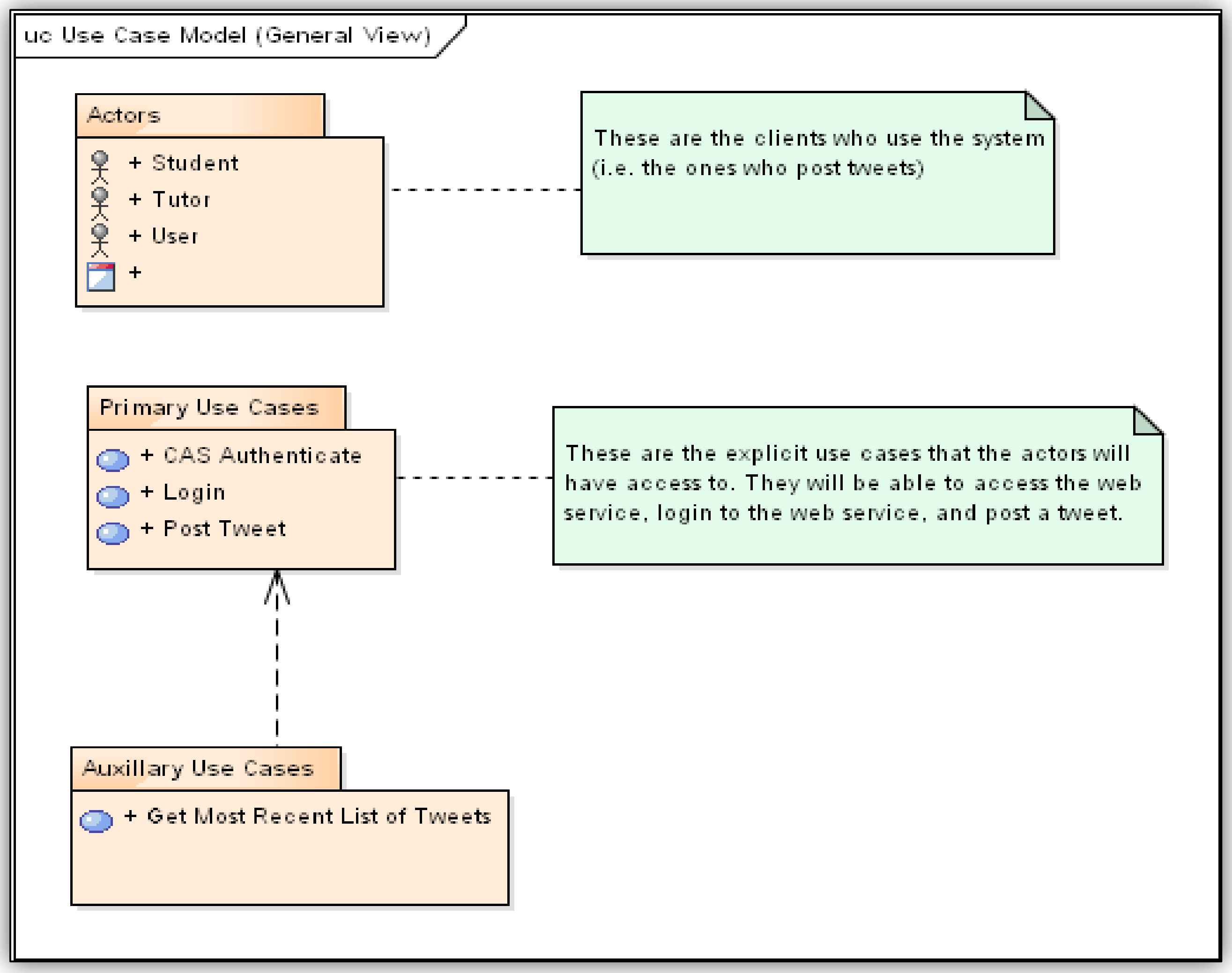

uc Use Case Model (General View)
Actors
+ Student
+ Tutor
+ User
+
These are the clients who use the system (i.e. the ones who post tweets)
Primary Use Cases
+ CAS Authenticate
+ Login
+ Post Tweet
These are the explicit use cases that the actors will have access to. They will be able to access the web service, login to the web service, and post a tweet.
Auxillary Use Cases
+ Get Most Recent List of Tweets

Figure 7 : Use Case Model (General View) - Iteration 2

4.1.2 Description

In the second iteration, the use case remains almost the same. We have added CAS Authentication to enhance security.

4.2 Primary Use Case View

4.2.1 Diagram

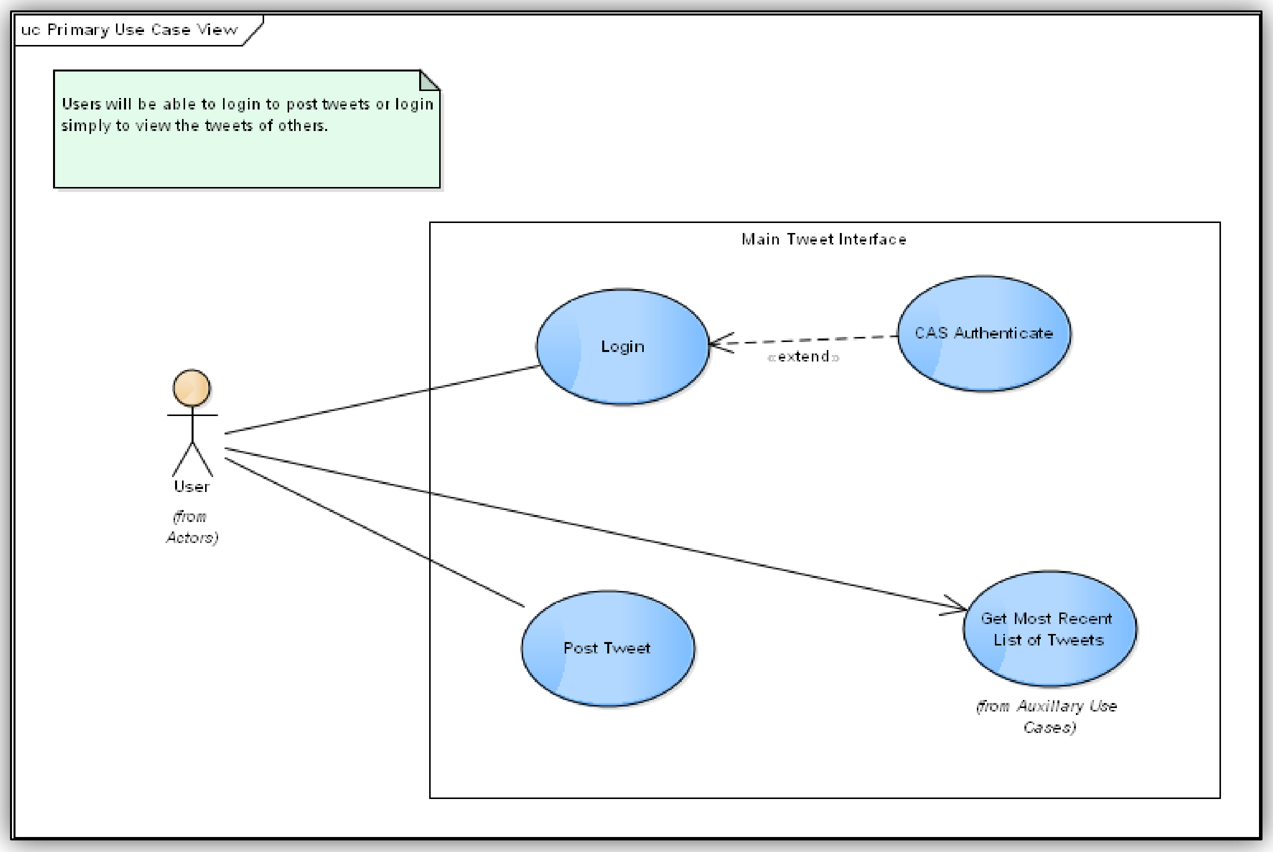

uc Primary Use Case View
Users will be able to login to post tweets or login simply to view the tweets of others.
Main Tweet Interface
Login
CAS Authenticate
«extend»
User
(from Actors)
Post Tweet
Get Most Recent List of Tweets
(from Auxillary Use Cases)

Figure 8 : Primary Use Case View - Iteration 2

4.2.2 Description

The primary use case also shows the CAS authentication which is used during login.

4.3 Auxillary Use Case View

4.3.1 Diagram

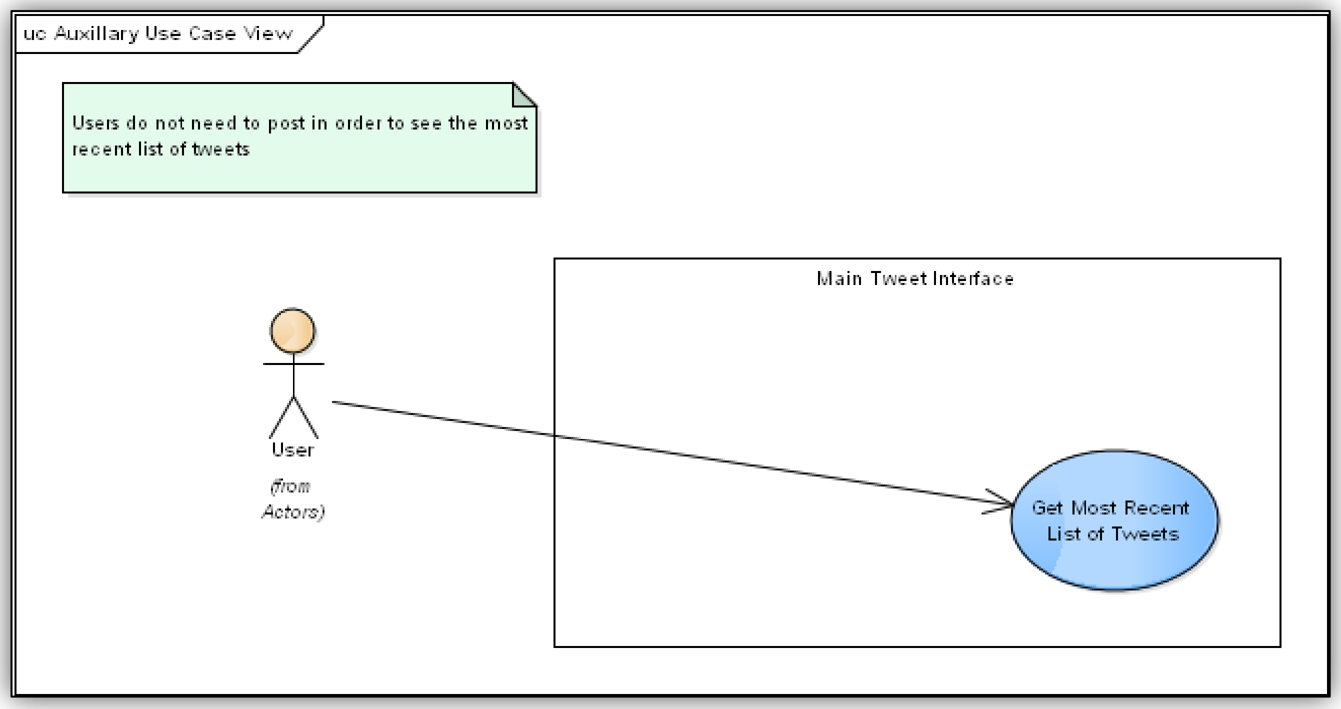

uc Auxillary Use Case View
Users do not need to post in order to see the most recent list of tweets
Main Tweet Interface
User
(from Actors)
Get Most Recent List of Tweets

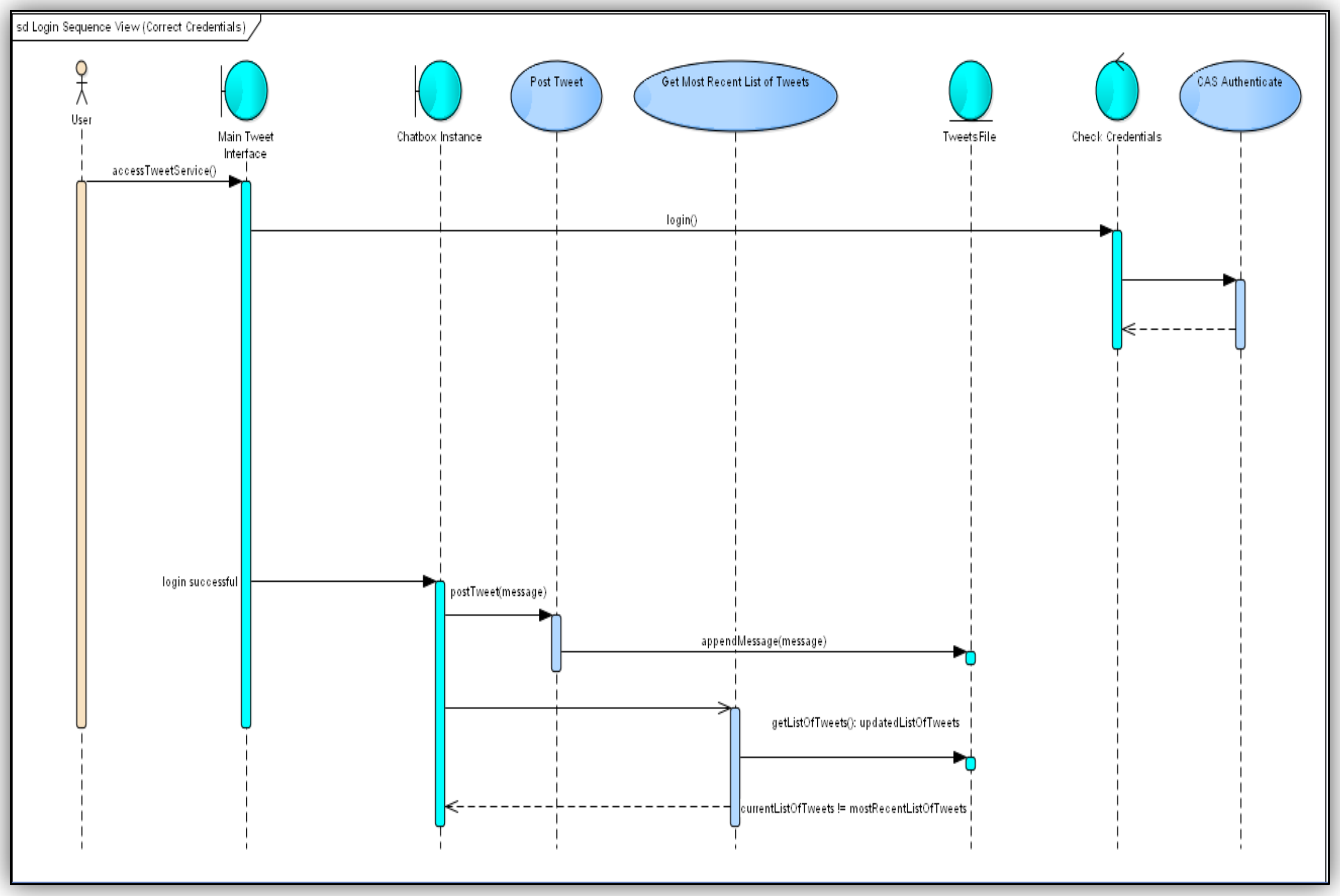
Figure 9 : Auxillary Use Case View - Iteration 2

4.3.2 Description

Here, the users do not need to post to see the most recent tweets.

4.4 Login Sequence View (Correct Credentials)

4.4.1 Diagram



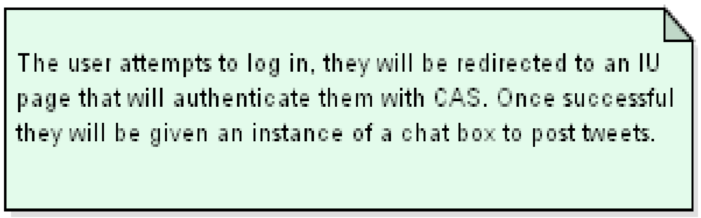

The user attempts to log in, they will be redirected to an IU page that will authenticate them with CAS. Once successful they will be given an instance of a chat box to post tweets.

Figure 10 : Login Sequence View (Correct Credentials) - Iteration 2

4.4.2 Description

1. The difference from the first iteration is that the user when he logs in, he has to go through the CAS authentication which is done by IUPUI and hence is very secure. When the CAS is

successful the user can log in to tweet or use the service. The retry count is maintained by the CAS authentication.

2. The user will get a list of most updated tweets when he logs into the system.

5 Detailed Software Design Description in Iteration 3

5.1 Use Case Model (General View)

5.1.1 Diagram

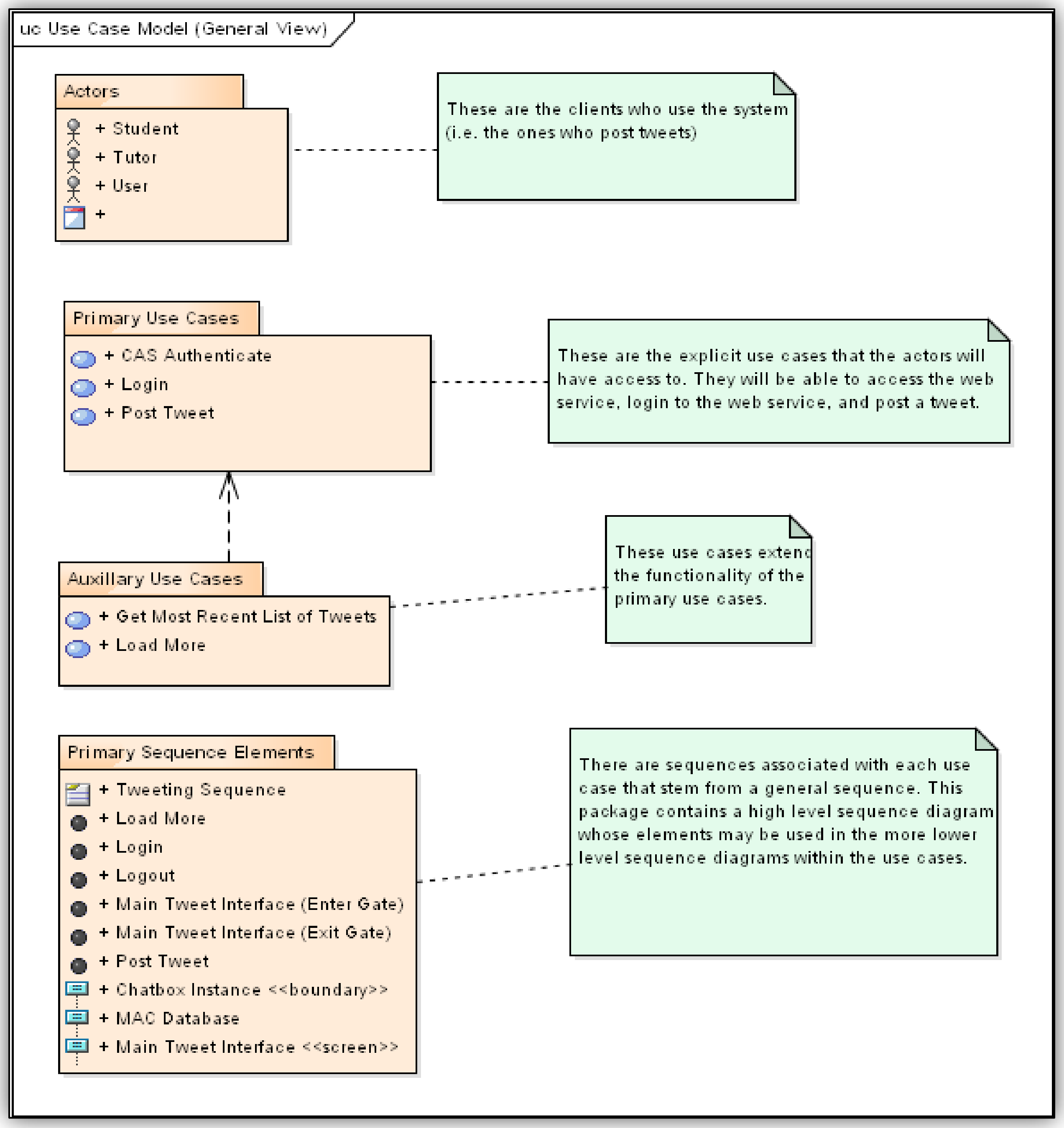

uc Use Case Model (General View)
Actors
+ Student
+ Tutor
+ User
+
These are the clients who use the system (i.e. the ones who post tweets)
Primary Use Cases
+ CAS Authenticate
+ Login
+ Post Tweet
These are the explicit use cases that the actors will have access to. They will be able to access the web service, login to the web service, and post a tweet.
Auxillary Use Cases
+ Get Most Recent List of Tweets
+ Load More
These use cases extend the functionality of the primary use cases.
Primary Sequence Elements
+ Tweeting Sequence
+ Load More
+ Login
+ Logout
+ Main Tweet Interface (Enter Gate)
+ Main Tweet Interface (Exit Gate)
+ Post Tweet
+ Chatbox Instance <<boundary>>
+ MAC Database
+ Main Tweet Interface <<screen>>
There are sequences associated with each use case that stem from a general sequence. This package contains a high level sequence diagram whose elements may be used in the more lower level sequence diagrams within the use cases.

Figure 11 : Use Case Model (General View) - Iteration 3

5.1.2 Description

1. Here, in Iteration 3 we have added the load more option which will load some previous tweets.

2. We have used MySQL for storing the tweet information.

3. The chatbox has a limit for the amount of characters to be sent.

4. The send button is also introduced to send out a tweet rather than pressing Enter.

5.2 Actors View

5.2.1 Diagram

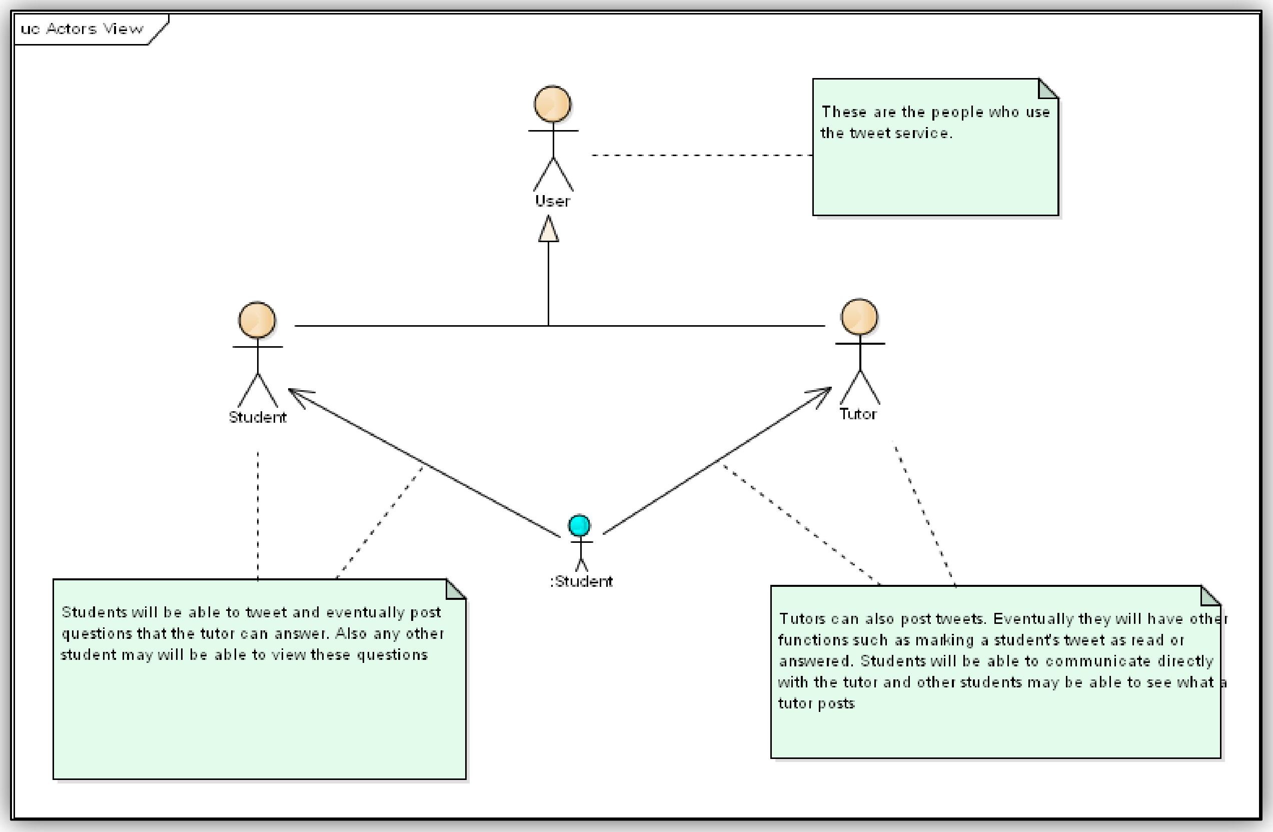

uc Actors View
User
These are the people who use the tweet service.
Student
Tutor
:Student
Students will be able to tweet and eventually post questions that the tutor can answer. Also any other student may will be able to view these questions
Tutors can also post tweets. Eventually they will have other functions such as marking a student's tweet as read or answered. Students will be able to communicate directly with the tutor and other students may be able to see what a tutor posts

Figure 12 : Actors View - Iteration 3

5.2.2 Description

This remains the same as Iteration 1.

5.3 Primary Use Case Sequence (General View)

5.3.1 Description

1. The user after login can post as many tweets as he wants and also can use the load more option to get the previous tweets on the screen.

2. Now, the tweets will be retrieved from MySQL Database.

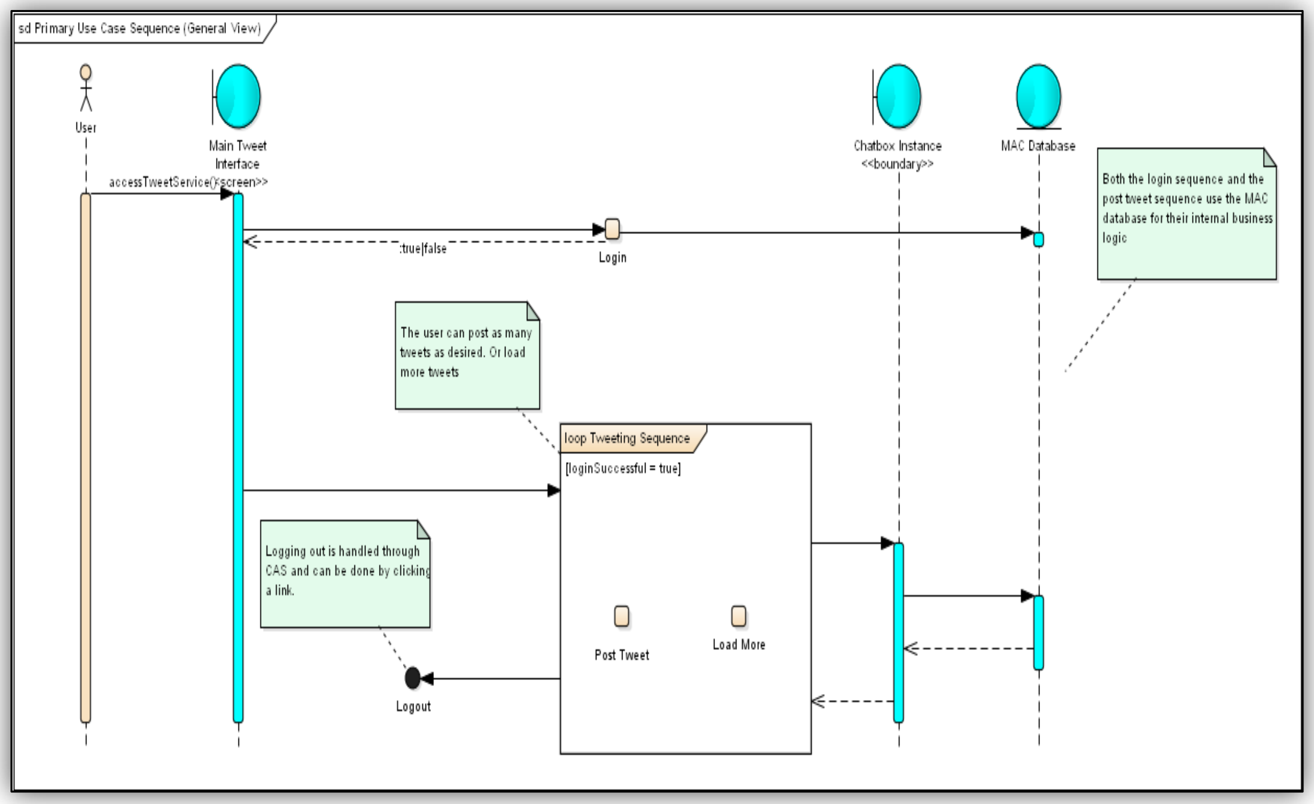

sd Primary Use Case Sequence (General View)
User
Main Tweet Interface
accessTweetService()<<screen>>
:true|false
Login
Chatbox Instance <<boundary>>
MAC Database
Both the login sequence and the post tweet sequence use the MAC database for their internal business logic
The user can post as many tweets as desired. Or load more tweets
loop Tweeting Sequence
[loginSuccessful = true]
Logging out is handled through CAS and can be done by clicking a link.
Post Tweet
Load More
Logout

Figure 13 : Primary Use Case Sequence (General View) - Iteration 3

5.4 Primary Use Case View

5.4.1 Description

This just shows the load more option which the user can use to load more previous tweets.

5.4.2 Diagram:

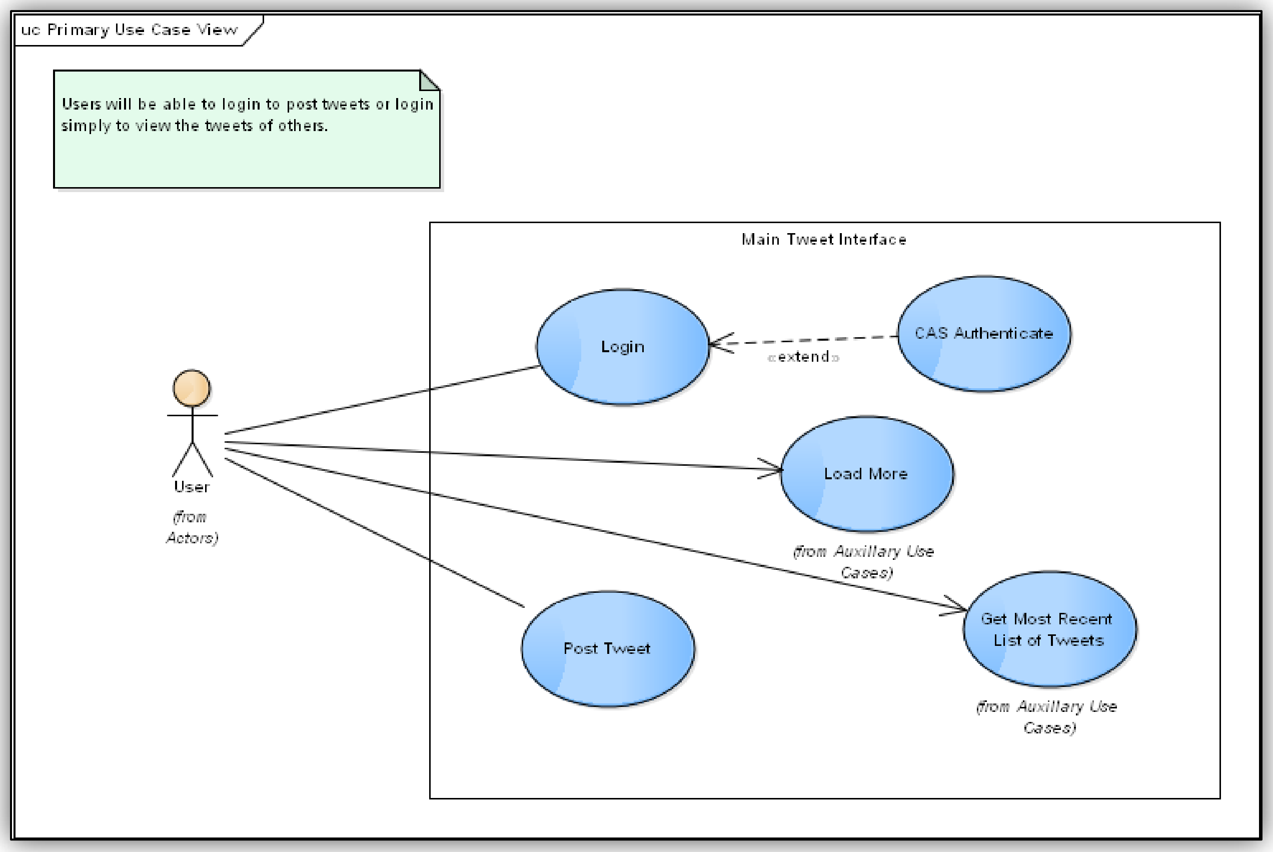

uc Primary Use Case View
Users will be able to login to post tweets or login simply to view the tweets of others.
Main Tweet Interface
Login
«extend»
CAS Authenticate
User
(from Actors)
Load More
(from Auxillary Use Cases)
Post Tweet
Get Most Recent List of Tweets
(from Auxillary Use Cases)

Figure 14 : Primary Use Case View - Iteration 3

5.5 Auxillary Use Case View

5.5.1 Description

It also shows the load more option which is used to retrieve more previous tweets.

5.5.2 Diagram

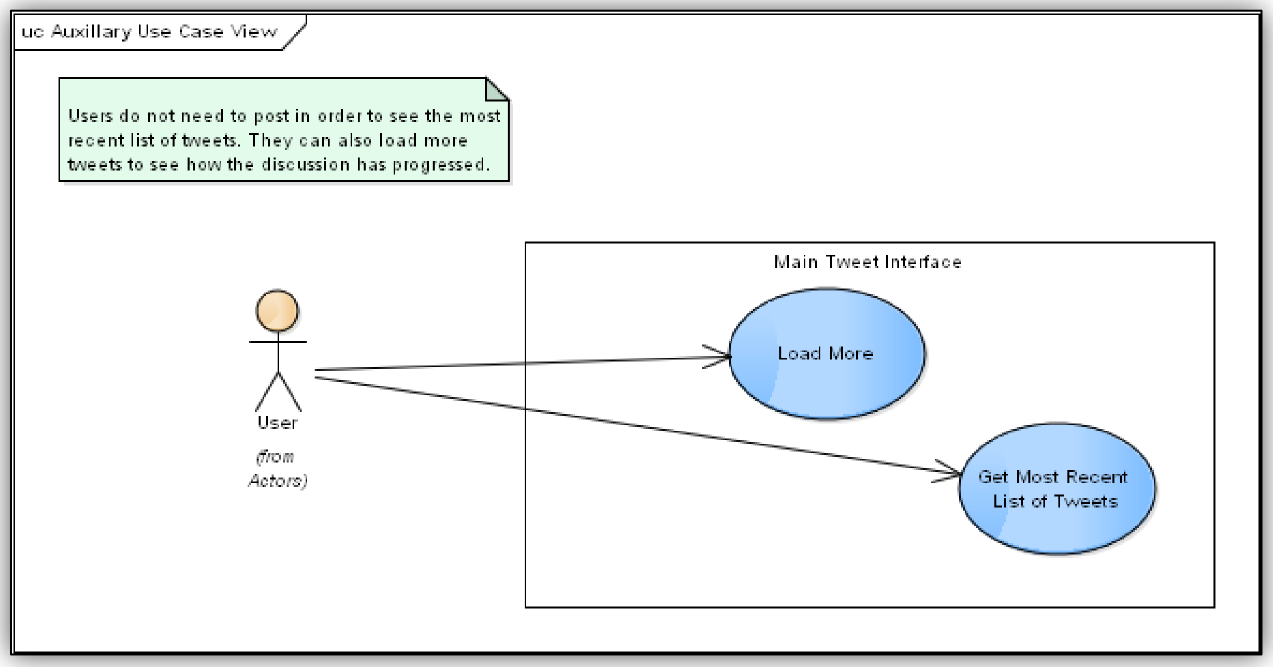

uc Auxillary Use Case View
Users do not need to post in order to see the most recent list of tweets. They can also load more tweets to see how the discussion has progressed.
Main Tweet Interface
Load More
Get Most Recent List of Tweets
User
(from Actors)

Figure 15 : Auxillary Use Case View - Iteration 3

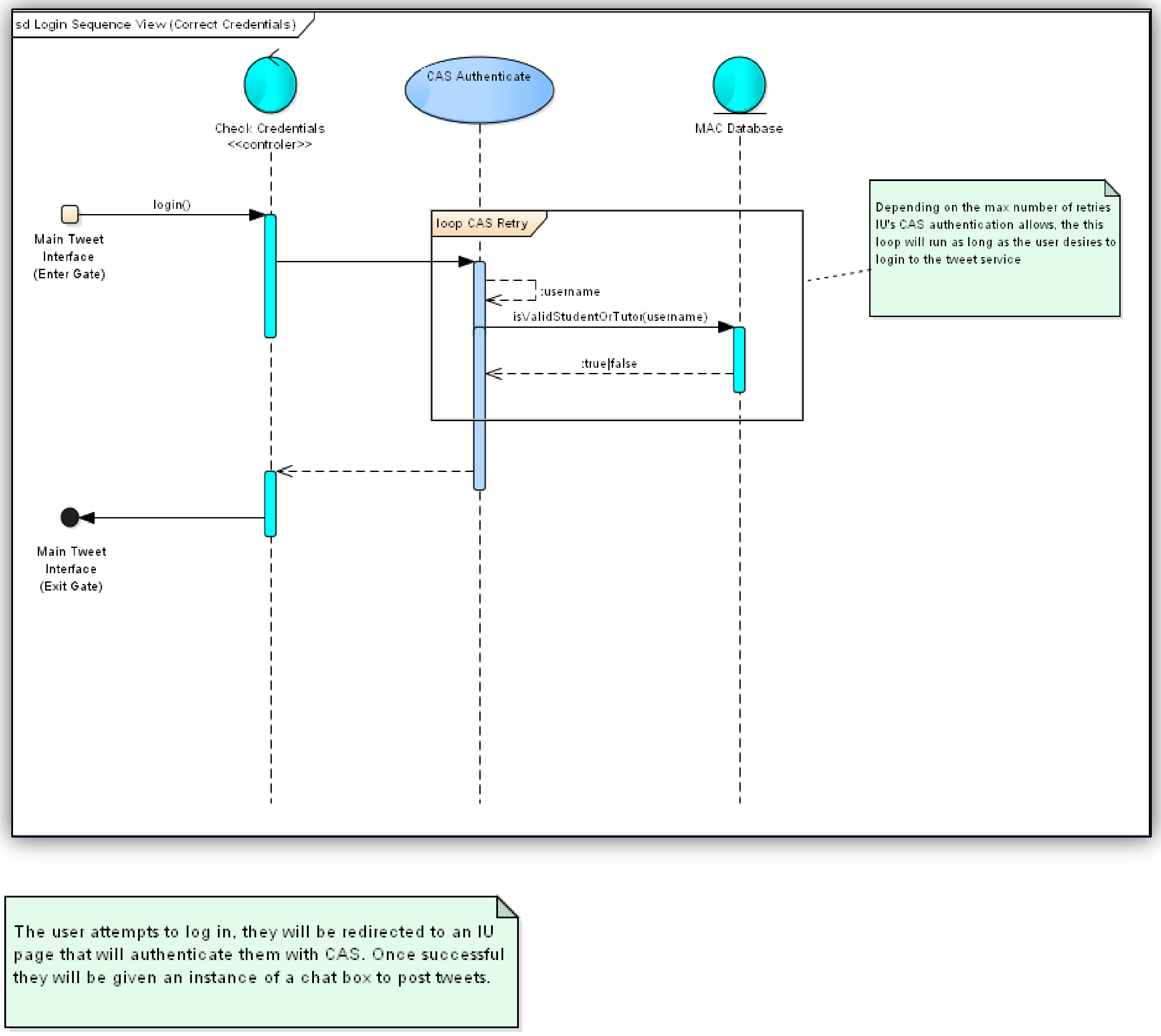
5.6 Login Sequence View

5.6.1 Description

1. This shows the login sequence controlled by the CAS authentication in which the retry is controlled by CAS and it is checked with the database.

2. The retry is allowed as long as CAS allows it. So, it has a very high level of security.

5.6.2 Diagram



Depending on the max number of retries IU's CAS authentication allows, the this loop will run as long as the user desires to login to the tweet service

The user attempts to log in, they will be redirected to an IU page that will authenticate them with CAS. Once successful they will be given an instance of a chat box to post tweets.

Figure 16 : Login Sequence View - Iteration 3

5.7 Post Tweet Sequence View

5.7.1 Description

1. Here, the user gets the list of most recent tweets.

2. Instead of being retrieved from the file, the tweets are retrieved from the MySQL server.

5.7.2 Diagram

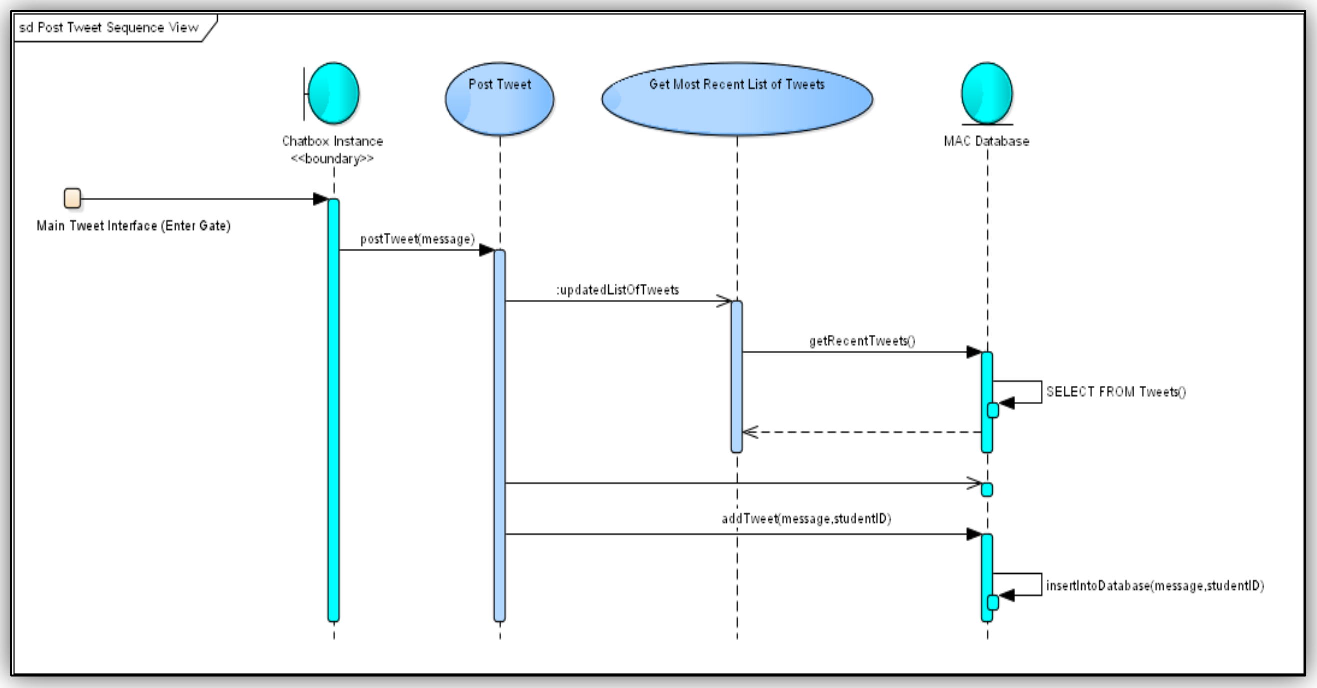

sd Post Tweet Sequence View
Chatbox Instance <<boundary>>
Post Tweet
Get Most Recent List of Tweets
MAC Database
Main Tweet Interface (Enter Gate)
postTweet(message)
:updatedListOfTweets
getRecentTweets()
SELECT FROM Tweets()
addTweet(message,studentID)
insertIntoDatabase(message,studentID)

Figure 17 : Post Tweet Sequence View - Iteration 3

5.8 Load More Sequence View

5.8.1 Description

1. In Iteration 3, the user can after tweeting use the load more option to retrieve the previous tweets for his convenience.

2. When the load more button is clicked then the list of previous tweets from the database is retrieved in backwards fashion and displayed on the screen.

3. The user can continue tweeting.

4. The chatbox has also a limit of sending messages.

5.8.2 Diagram

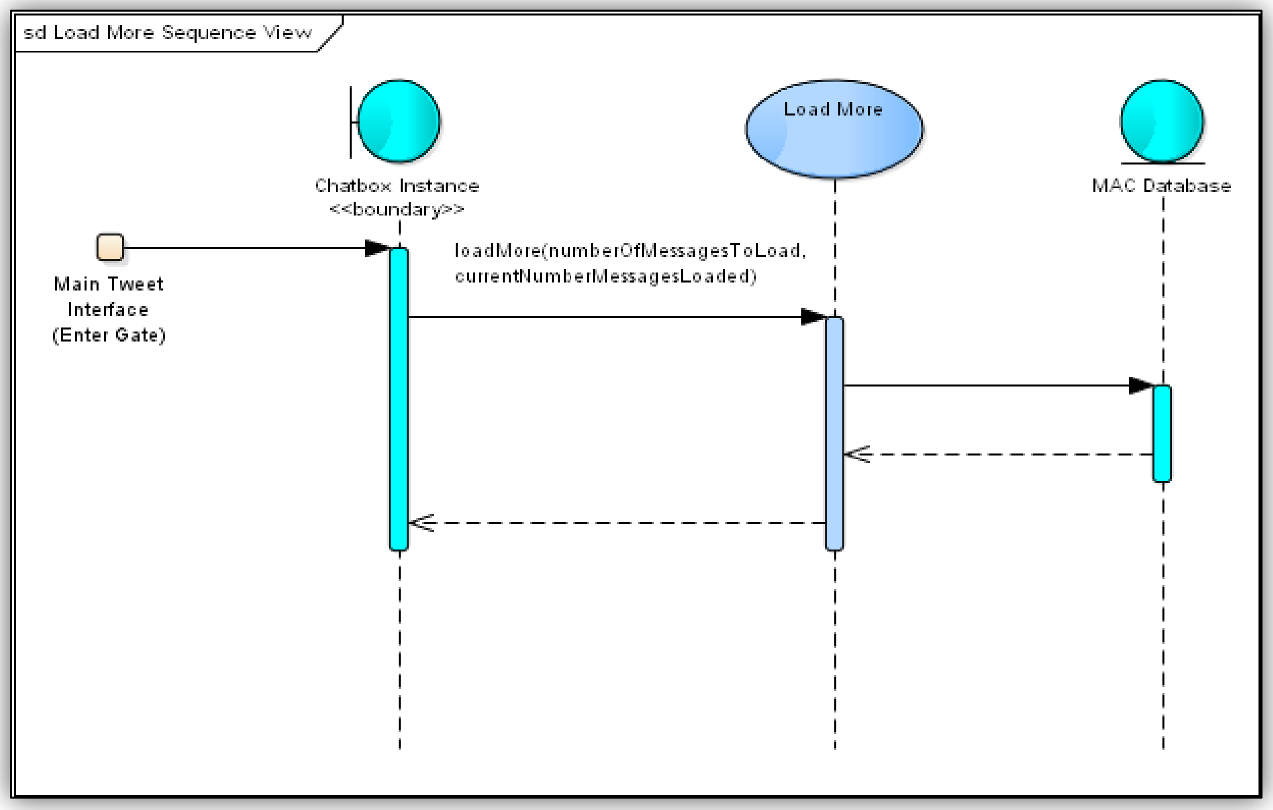

sd Load More Sequence View
Chatbox Instance
<<boundary>>
Load More
MAC Database
Main Tweet Interface (Enter Gate)
loadMore(numberOfMessagesToLoad, currentNumberMessagesLoaded)

Figure 18 : Load More Sequence View - Iteration 3

5.9 Database

5.9.1 Description

1. This shows the description of the database how the tables are maintained and the basic structure.

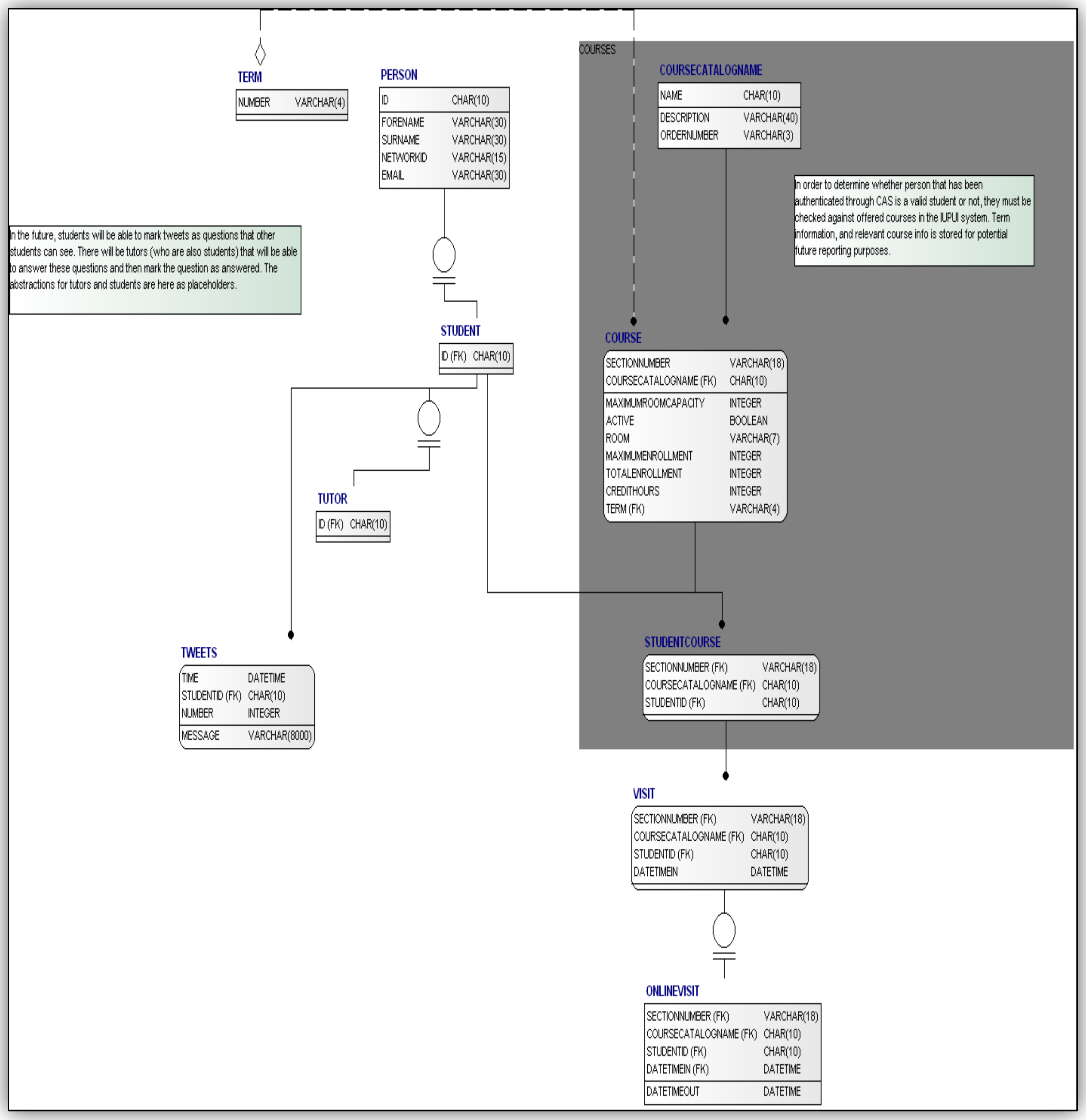
2. The students have ID from their IUPUI ID.

3. The Tweets are stored in message field and they also have date and time tweeted with the student ID who tweeted it.

4. The course part is handled by CAS and the login, username and password details are stored and retrieved by CAS Authentication.

5. Hence, the sensitive data of usernames and passwords are handled by CAS so security has increased.

5.9.2 Diagram



In order to determine whether person that has been authenticated through CAS is a valid student or not, they must be checked against offered courses in the IUPUI system. Term information, and relevant course info is stored for potential future reporting purposes.

In the future, students will be able to mark tweets as questions that other students can see. There will be tutors (who are also students) that will be able to answer these questions and then mark the question as answered. The abstractions for tutors and students are here as placeholders.

Figure 19 : Database - Iteration 3

6 Detailed Software Design Description in Iteration 4

6.1 Use Case Model (General View)

6.1.1 Diagram

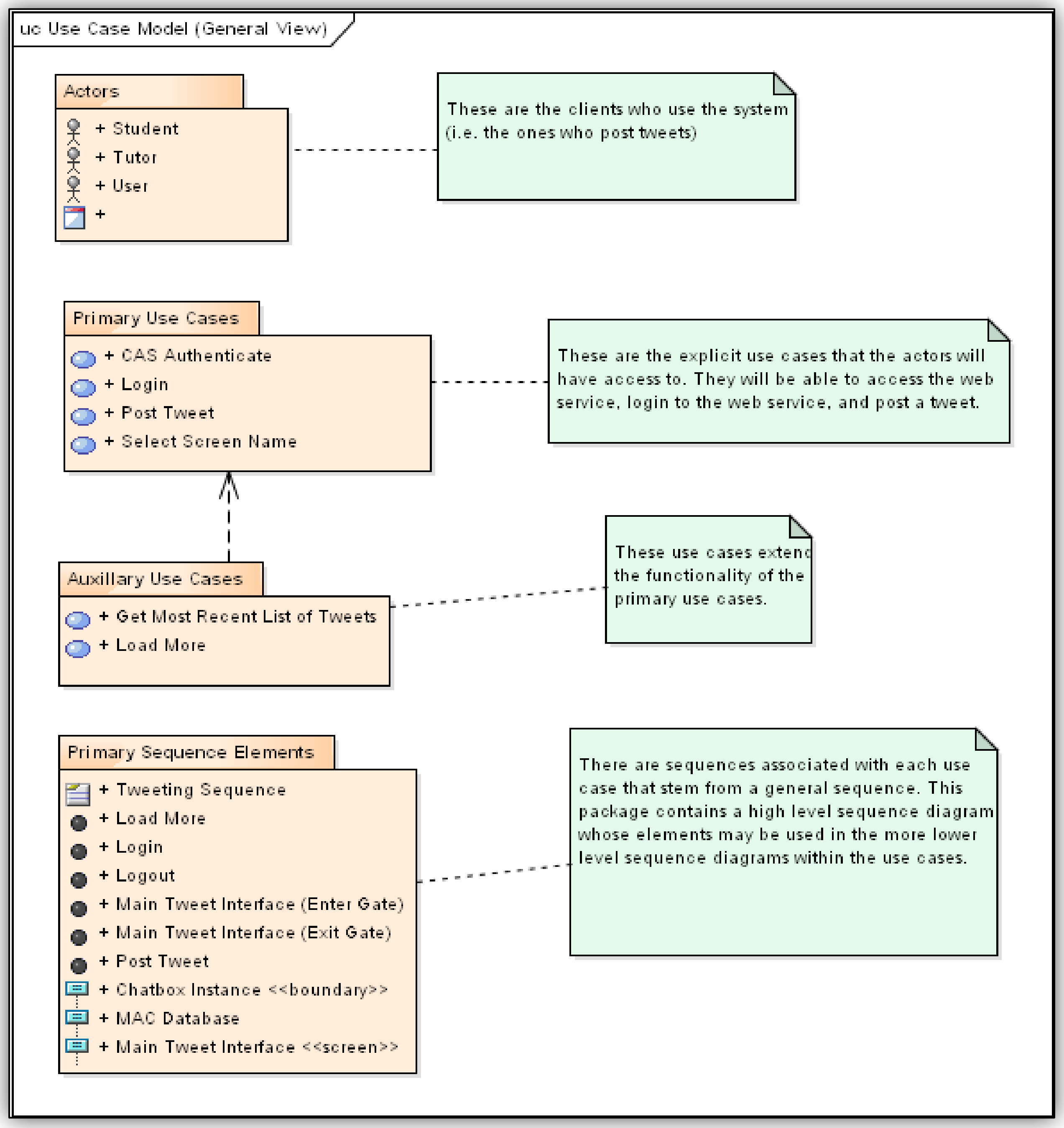

uc Use Case Model (General View)
Actors
+ Student
+ Tutor
+ User
+
These are the clients who use the system (i.e. the ones who post tweets)
Primary Use Cases
+ CAS Authenticate
+ Login
+ Post Tweet
+ Select Screen Name
These are the explicit use cases that the actors will have access to. They will be able to access the web service, login to the web service, and post a tweet.
Auxillary Use Cases
+ Get Most Recent List of Tweets
+ Load More
These use cases extend the functionality of the primary use cases.
Primary Sequence Elements
+ Tweeting Sequence
+ Load More
+ Login
+ Logout
+ Main Tweet Interface (Enter Gate)
+ Main Tweet Interface (Exit Gate)
+ Post Tweet
+ Chatbox Instance <<boundary>>
+ MAC Database
+ Main Tweet Interface <<screen>>
There are sequences associated with each use case that stem from a general sequence. This package contains a high level sequence diagram whose elements may be used in the more lower level sequence diagrams within the use cases.

Figure 20 : Use Case Model (General View) - Iteration 4

6.1.2 Description

1. In Iteration 4, the changes made are that the website is deployed onto Amazon Web Services (AWS) as per the software requirements and the users are now using the website hosted on AWS.

2. The users also have the ability to change screen names which will be reflected in their tweets.

3. But in the main database the tweets are stored as their real user names and the screen names are also tracked to ensure protection against misbehavior.

4. Timestamps are also added as now we track screen names.

6.2 Primary Use Case View

6.2.1 Diagram

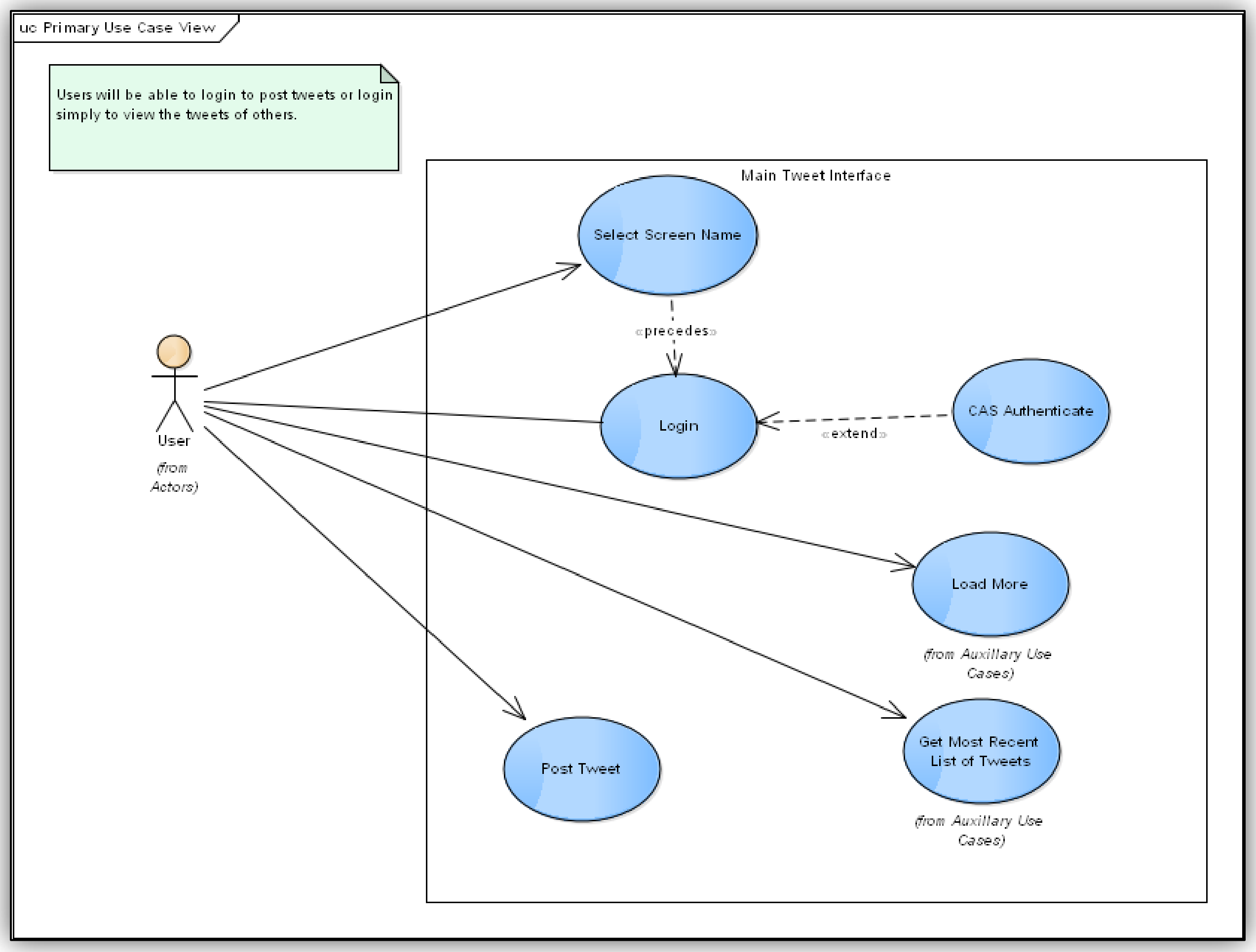

uc Primary Use Case View
Users will be able to login to post tweets or login simply to view the tweets of others.
Main Tweet Interface
Select Screen Name
«precedes»
Login
«extend»
CAS Authenticate
User
(from Actors)
Load More
(from Auxillary Use Cases)
Get Most Recent List of Tweets
(from Auxillary Use Cases)
Post Tweet

Figure 21 : Primary Use Case View - Iteration 4

6.2.2 Description

1. The only change in the primary use case is that the select screen name element is introduced and the user can pick his screen name and tweet if he does not want to use the IUPUI user name.

2. The rest of the functionalities remain the same.

6.3 Primary Use Case Sequence (General View)

6.3.1 Diagram

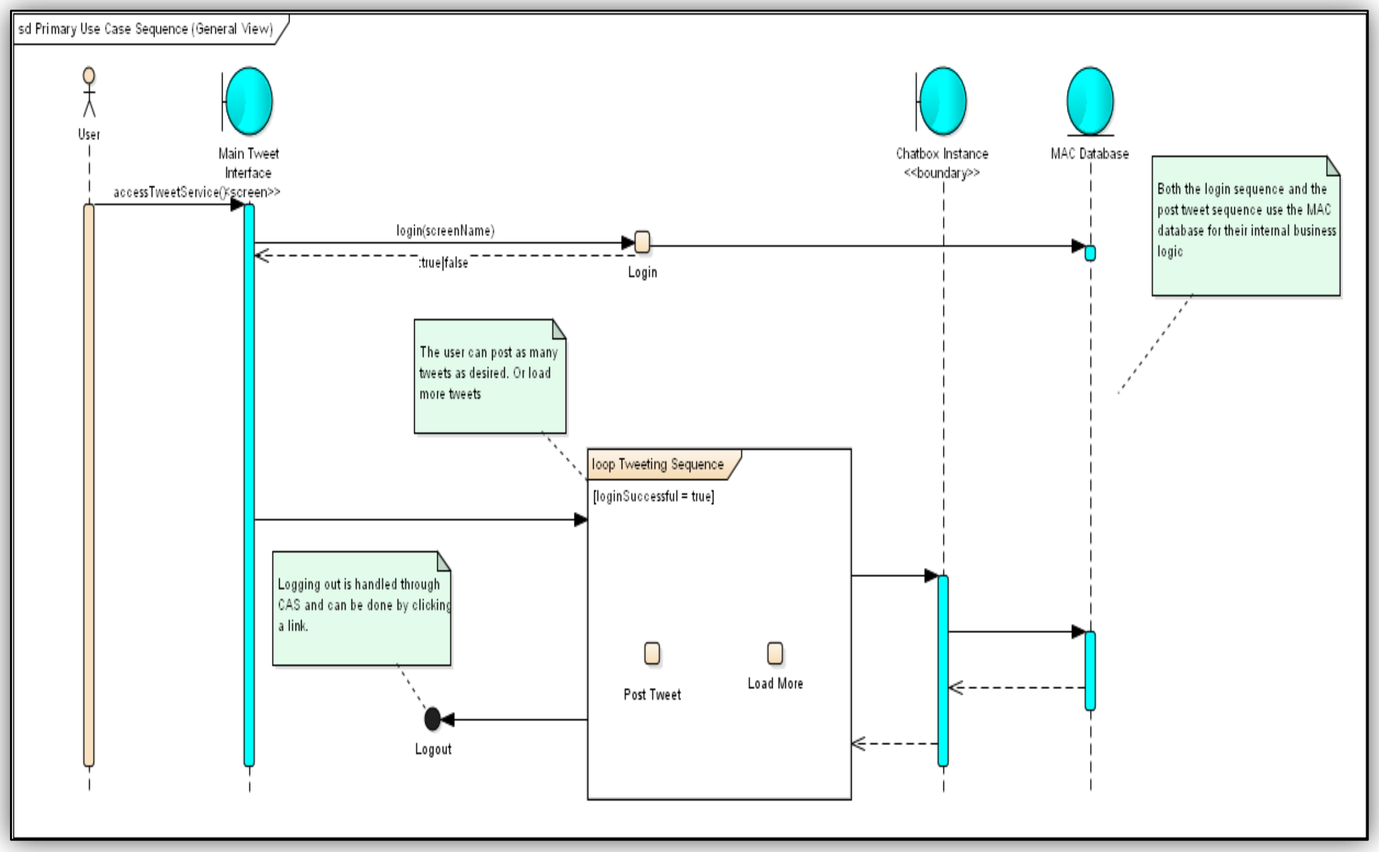

sd Primary Use Case Sequence (General View)
User
Main Tweet Interface
Chatbox Instance <<boundary>>
MAC Database
accessTweetService()<<screen>>
login(screenName)
:true|false
Login
Both the login sequence and the post tweet sequence use the MAC database for their internal business logic
The user can post as many tweets as desired. Or load more tweets
loop Tweeting Sequence
[loginSuccessful = true]
Logging out is handled through CAS and can be done by clicking a link.
Post Tweet
Load More
Logout

Figure 22 : Primary Use Case Sequence (General View) - Iteration 4

6.3.2 Description

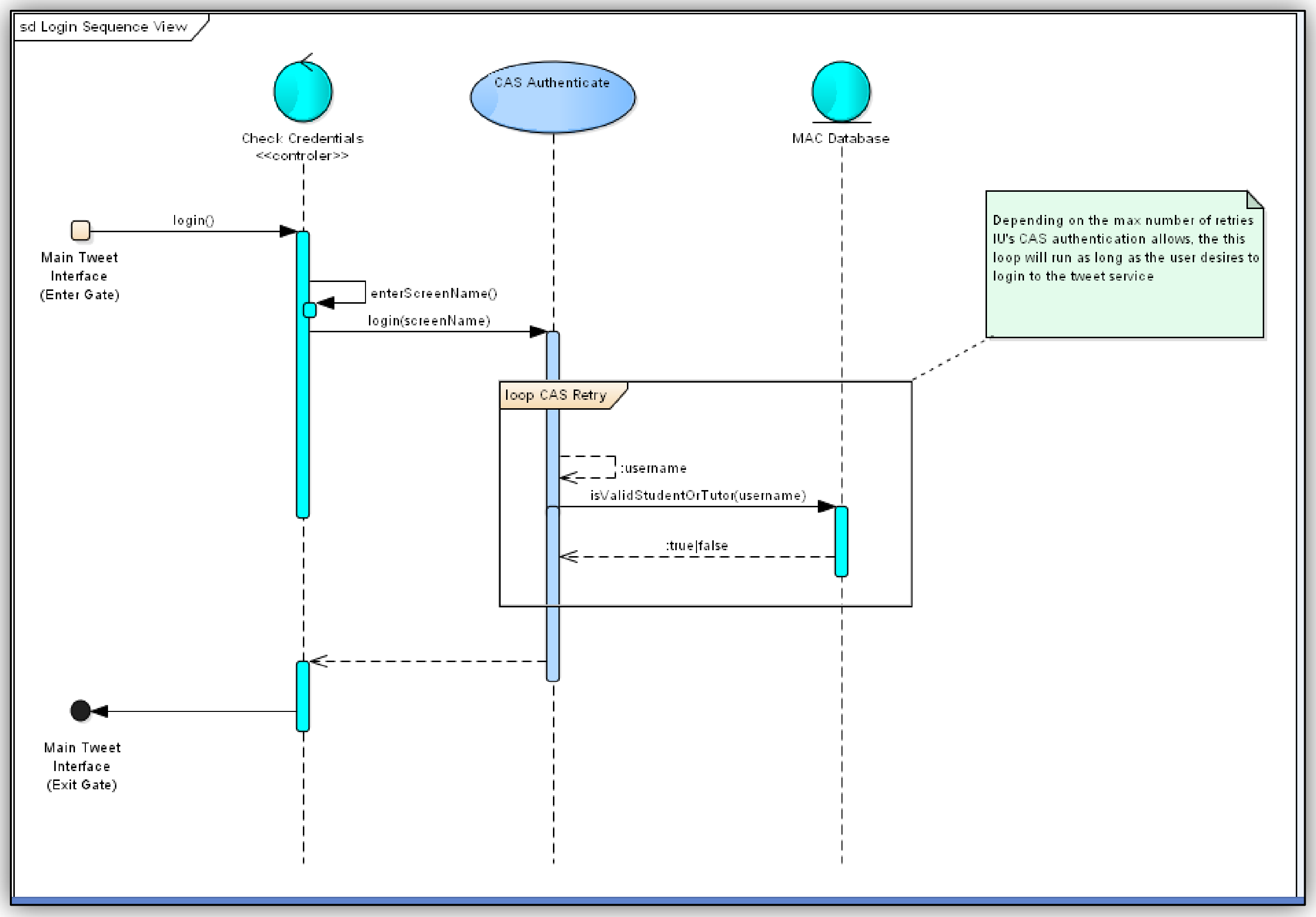
1. In this the user can choose a screen name of his choice and then login and post messages.

2. But it will show his screen name and not the IUPUI user name.

3. The credentials are appropriately checked with the database using CAS authentication.

6.4 Login Sequence View

6.4.1 Diagram:



The user attempts to log in, they will be redirected to an IU page that will authenticate them with CAS. Once successful they will be given an instance of a chat box to post tweets.

Figure 23 : Login Sequence View - Iteration 4

6.4.2 Description

1. The user goes to the main tweet interface and then chooses a user name and tries to log in using IUPUI credentials at the CAS authentication.

2.If successful, he will be able to tweet and his screen name will appear.

3. If unsuccessful, he will be allowed to retry as many times as CAS authentication allows.

6.5 Load More Sequence View

6.5.1 Diagram

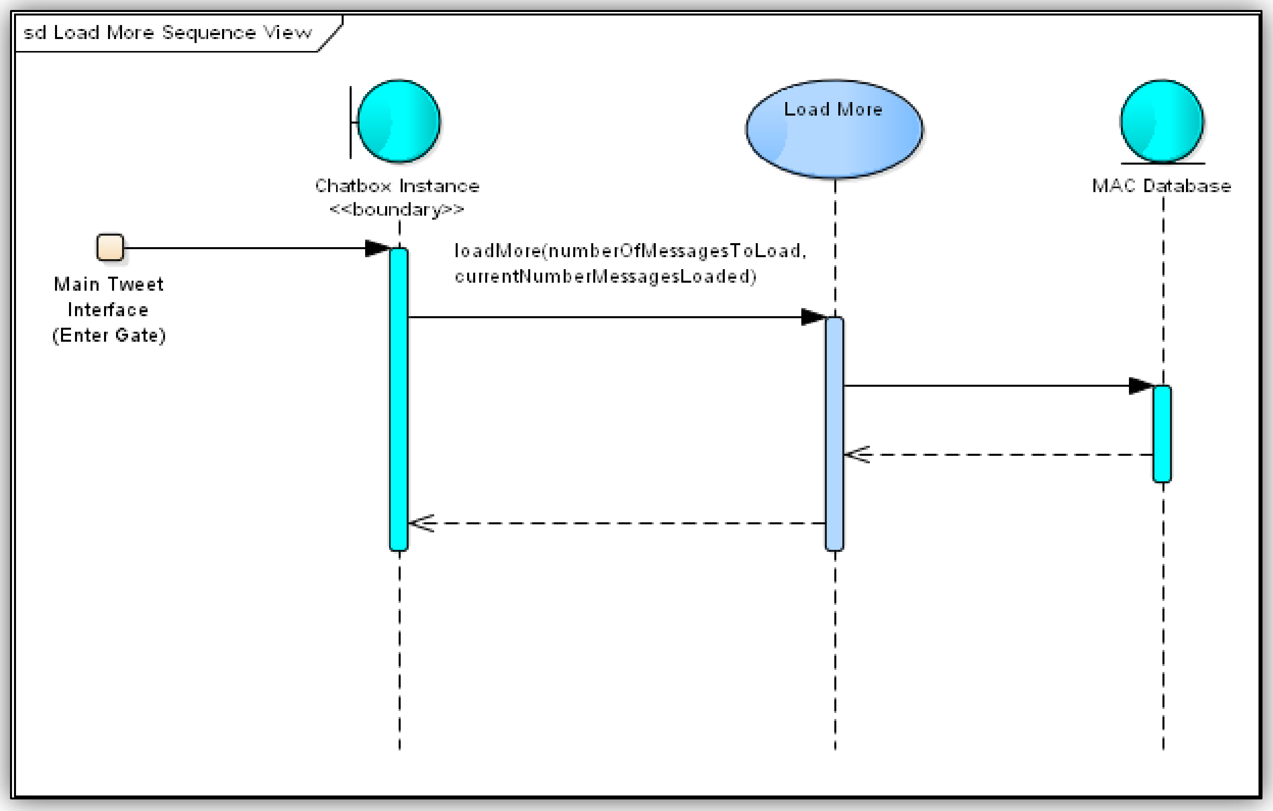

sd Load More Sequence View
Chatbox Instance
<<boundary>>
Load More
MAC Database
Main Tweet Interface (Enter Gate)
loadMore(numberOfMessagesToLoad, currentNumberMessagesLoaded)

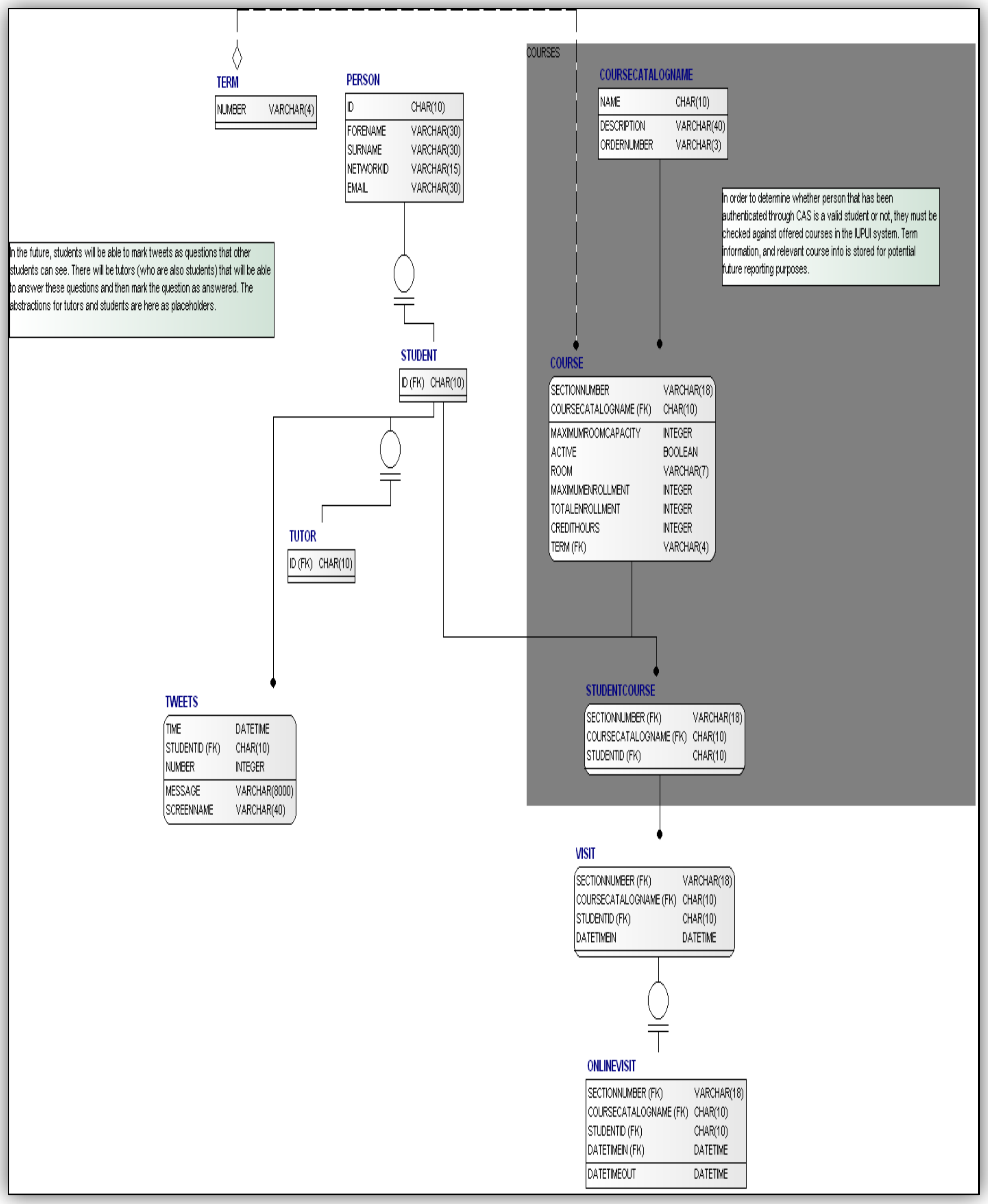
Figure 24 : Load More Sequence View - Iteration 4

6.5.2 Description

In this, when the user selects the load more option it will load more previous tweets and their associated screen names.

6.6 Database

6.6.1 Diagram



In order to determine whether person that has been authenticated through CAS is a valid student or not, they must be checked against offered courses in the IUPUI system. Term information, and relevant course info is stored for potential future reporting purposes.

In the future, students will be able to mark tweets as questions that other students can see. There will be tutors (who are also students) that will be able to answer these questions and then mark the question as answered. The abstractions for tutors and students are here as placeholders.

Figure 25 : Database - Iteration 4

6.6.2 Description

1. The database structure remains the same and it is deployed on AWS.

2. The screen name is added to the tweets table and hence when asked is retrieved from the table or on being created is stored.

3. The timestamps associated with the tweets are also stored as per the screen names.

7 Software Architecture

Three styles of software have been used in our software. They are as follows:

7.1 Client-Server Architecture

The software has a client server style with 3 layers i.e., the presentation layer, business logic layer and database layer.

7.1.1 Presentation Layer:

It is the GUI of the software system. The client or the user interacts with it to perform his tweeting activities with the server.

7.1.2 Business Logic Layer:

It is the middle layer where the processing engine works for querying to the database layer and retrieving the results and putting it on the application GUI.

7.1.3 Database Layer:

This is where the data resides and the business logic layer queries it depending upon the instructions provided by the GUI or presentation layer.

7.2 Repository Architecture

The software is centered around data that is tweeting information which it does by sending tweets, loading tweets and storing the tweets. Hence it follows a repository architecture.

7.3 Parallel Communicating Processes Architecture

When a user logs in, a separate session is created for that user. Then as per the environment of the software usage in parallel the users get updates when others add a tweet and the user can do his own operations. Hence, it is a parallel communicating processes architecture.

8 References

1. *Software Engineering Modern Approaches Second Edition* Eric J. Braude and Michael E. Bernstein, Wiley Publications.

2. B. W. Boehm, J. R. Brown, M. Lipow, *Quantitative Evaluation of Software Quality*, TRW Systems and Energy Group, (1976).