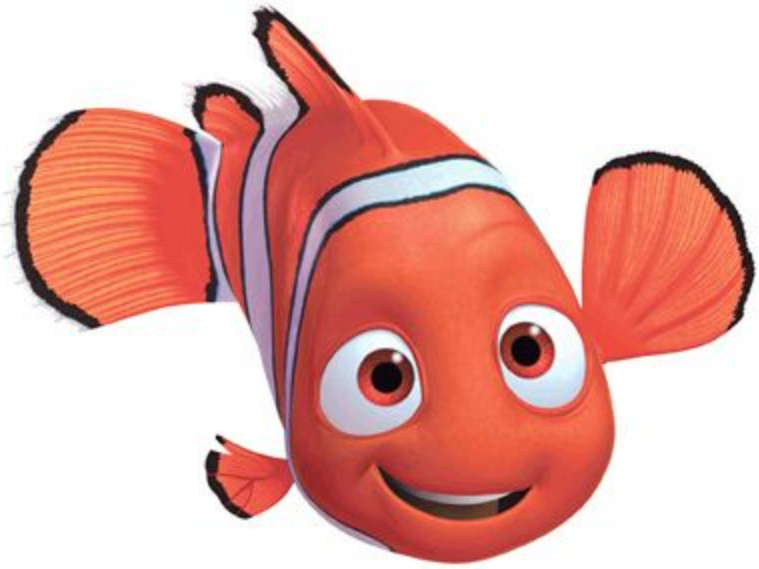
KU MEDICAL CENTER The University of Kansas

TENNESSEE TECH UNIVERSITY

# Never-Ending Medical Operative (NEMO)



**User Requirements Document**
**Version: 1.2**

**Authors:**
Cody Moffitt
Jackson Chandler
Andrew Pachuilo
Oliver Payne

# Table of Contents

# Chapter 1: Introduction

## 1.01: About the project

The goal of this project is to create a never-ending healthcare informatics learner. An artificial intelligence agent will continually be applying hybrid data mining approaches to medical data for a specific question posed by domain experts. The agent will learn from domain experts in the form of a feedback mechanism, or loop, to improve its predictions, as well as its learning ability. Domain experts will be able to pose questions and provide feedback to the learner via a web application. The learner will interface with various machine learning algorithms to form predictions based on the data. The learner and web application will use a subset of data created from the existing University of Kansas Medical Center database and implement secure technologies to protect patient confidentiality and conforming HIPAA regulations.

## 1.02: System Overview

The NEMO system will consist of several distinct parts:

- **Data Loader**
  - The data loader will fetch and consolidate medical data from the target i2b2 database for storage in the NEMO data mart.
  - The data loader will be able to run manually or on a schedule.
- **Data Mart**
  - The data mart will hold a subset of the medical data from the KUMC i2b2 database for use by the AI and web application.
  - The data mart will also hold non-medical information relevant to the AI, such as user questions and feedback.
- **AI Controller**
  - The AI controller will continuously spawn AI nodes to process user questions.
- **Web Application**
  - The web client will have a secure login for users.
  - The web client will allow users to view, create, delete, and edit questions as well as observe answers and provide feedback to refine the AI learners.
  - Users will be able to view and duplicate other users' questions via a global dashboard.

# 1.03: Glossary

**NEMO**: Never-Ending Medical Operative. The project is based off of the Never-Ending Language Learner (NELL) system. The goal is to create a system that continually learns on real-world data in order to make predictions and answer medical related questions.
**Data Mart**: A smaller subset of the medical database. Entries here will be partitioned into test data and learner data. The learners will use this data to answer questions and form conclusions.
**Database**: A collection of a large amount of data. This data is organized into entries of records. Each record is made up of several components.
**i2b2**: A National Center for Biomedical Computing which developed a biomedical informatics framework funded by the U.S. National Institutes of Health.
**AI**: Artificial intelligence. It is the process of simulating one or more human characteristics with a computer. In this particular case, the artificial intelligence is machine learning.
**Classifier Algorithm**: An algorithm that is purposed with evaluating data and new classifications.
**AI Node**: A particular instance of AI algorithms.
**Learner**: An algorithm that is purposed with evaluating data and forming conclusions based off of it. A machine learning algorithm.
**HIPAA**: Health Insurance Portability and Accountability Act.
**KUMC**: Kansas University Medical Center.
**Soft Edit**: An edit of a question that creates a new question in the database.
**Hard Edit**: An edit of a question that changes the original question in the database without creating a new one.
**Global Dashboard:** This is the area of the web client where users can view other users' questions.
**User Dashboard:** This is the area of the web client where users can view only their own questions.

# Chapter 2: User Scenarios

---

## 2.01: User logs into NEMO web application

**Author**: Oliver Payne
**Initial Assumptions**:

- Web server is up and running
- The account database is up and able to validate credentials

**Normal Flow**:

User navigates to the login screen of the online webpage. User enters his or her username and password to connect to the application. If the credentials are invalid, the user is informed and prompted to re-enter them. The user enters the correct credentials. The account database authenticates their login and grants them access to the web application.

**What Can Go Wrong**:

- The account database could have a problem authenticating the user. The user is informed that there is a problem with the server and he or she is instructed to contact the system administrator
- The user may not have an account set up yet. The use is instructed to contact the system administrator to set up an account.

**Concurrent Events**:

- Other users will be logging in to the web application.

**State When Scenario Finishes**:

User is successfully logged in to the web application. User is shown a menu of different services that he or she can navigate to.

# 2.02: User poses question via web application

**Author**: Cody Moffitt
**Initial Assumptions**:

- User is logged into a valid session.
- The database is up and running.

**Normal Flow**:

The user navigates to the Question page of the web application. The user creates their question with the drop down menus, then chooses from a list of narrowing attributes. The user fills out the parameters they desire for the narrowing attributes they've chosen.

The user saves this question to the database, by clicking the submit button.

**What Can Go Wrong**:

- The submission to the database may timeout. The user is then informed that the question was not submitted and that he or she should check with the system administrator if the problem persists.
- The user may submit a question that is a duplicate of another question. The user is informed when this happens.
- The user's login session may timeout if they take too long. The user is then prompted to renew the session by re-entering his or her credentials.

**Concurrent Events**:

- Other users may be entering/editing questions at the same time.
- The AI will be processing existing questions in the database.

**State When Scenario Finishes**:

The database will now have the user's question saved for processing. The question will have a status of "Queued" until processed. The client web page will indicate the question has been saved to the database.

**Notes**:

On the Question page the user may pose a new question for the AI to process. The user is able to create the question via a series of drop-down menus and other varying input dialogs for parameters.

For example, if the user wishes to pose the question, "What is the likelihood of readmittance for patients with glucose levels between 70 and 90 mg/dL?" The user would select "Likelihood of" in the first dropbox, "readmittance for" in the second dropbox, and choose "with glucose levels" from a predefined list of "with" attributes. The drop-down menus and plain text words will form the complete question on screen.

After the attribute is selected, inputs for additional parameters will be shown. These additional parameters will be dependent on the narrowing attribute(s) chosen. The user will also be able to select additional narrowing attributes, so they could pose a question like, "What is the likelihood of readmittance for patients with glucose levels between 70 and 90 mg/dL and BMI between 18 and 25 $kg/m^2$?"

## 2.03: User deletes a question

**Author**: Jackson Chandler
**Initial Assumptions**:

- User is logged into a valid session.
- The database is up and running.

**Normal Flow**:

User navigates to the question page, where questions can be asked and past questions are displayed along with their status. The user clicks the option “delete” to remove the question. The question is removed, but any data will be kept for further use.

**What Can Go Wrong**:

- Database fails to delete the question the user is notified.
- The question is deleted but is still displayed on the user’s “questions” dashboard, the user should contact the administrator.

**Concurrent Events**:

- Other users may be deleting questions at the same time.
- The AI will be processing existing questions in the database.

**State When Scenario Finishes**:

The question is deleted from the database. Once successful then the page will notify the user that the question has been deleted.

# 2.04: User does a hard edit of question

**Author**: Oliver Payne
**Initial Assumptions**:

- The user is logged in to a valid session
- The question database is up and running.
- The original question has already been created.

**Normal Flow**:

The user navigates to the Question page on the web client, and is presented with a list of questions he or she has previously asked. The user selects a particular question, which displays options for the submission. The web page then displays the question creation page, with each drop-down menu filled out with the information from the original submission. At this point, the user is adjusts the original question as needed. When he or she is finished, the user presses the “Submit” button. After pressing the button, the user is asked if he or she wants to “Create a New Question” or “Edit the Original Question”. The user selects “Edit the Original Question” and then confirms the submission.

**What Can Go Wrong**:

- The submission to the database may timeout. The user is then informed that the question was not submitted and that he or she should check with the system administrator if the problem persists.
- The user may submit a question that is a duplicate of another question. The user is informed when this happens.
- The user’s login session may timeout if they take too long. The user is then prompted to renew the session by re-entering his or her credentials.

**Concurrent Events**:

- Other users may be editing questions at the same time.
- The AI will be processing existing questions in the database.

**State When Scenario Finishes**:

The question is submitted to the AI database, where it will stayed queued until it is processed. The web application will display a message stating that the question has been submitted to the database. The question will wait to begin processing until after the original question has finished and feedback for it has been given. During this time, the question listing on the web application side will display “Queued” under its status.

## 2.05: User does a soft edit of question

**Author**: Cody Moffitt
**Initial Assumptions**:

- User is logged into a valid session.
- The database is up and running.
- The original question has already been created.

**Normal Flow**:

The user navigates to the question page on the web application, and is presented with a list of questions they have previously asked, and options for each of those questions. The user selects edit on a specific question. The web page presents the same view that is shown while posing a new question, but with the drop-down menus and various input dialog already filled out to the options selected initially for that question. The user edits the drop-down menus and inputs.

The user clicks “Submit” when they are finished editing the question. They are then given the option to save it as a new question, or replace the original question. The user chooses to save it as a new question.

**What Can Go Wrong**:

- The submission to the database may timeout. The user is then informed that the question was not submitted and that he or she should check with the system administrator if the problem persists.
- The user may submit a question that they have already posed. The user is informed when this happens.
- The user’s login session may timeout if they take too long. The user is then prompted to renew the session by re-entering his or her credentials.

**Concurrent Events**:

- Other users may be entering/editing questions at the same time.
- The AI will be processing existing questions in the database.

**State When Scenario Finishes**:

The database will now have the user’s question saved for processing. The question will have a status of “Queued” until processed. The client web page will indicate the question has been saved to the database.

## 2.06: User gives feedback to the AI from processed question

**Author**: Andrew Pachuilo
**Initial Assumptions**:

- The user is logged in to a valid session.
- The database is running and has processed questions that need feedback from user.

**Normal Flow**:

While on the User Dashboard the user looks over their previously asked questions. They see one of their questions has been processed. After selecting the question, the user is able to give feedback on the AI's response by selecting either 'right' or 'wrong'. They notice that the AI performed poorly on the test data so they click on the wrong button for the AI's feedback. This feedback is then stored in the database for further use by the AI.

**What Can Go Wrong**:

- The user may accidently hit the wrong button when giving feedback. The user will be given a grace period to undo their action via a temporary on page 'undo' button that will pop up shortly after they click 'right' or 'wrong'.
- The database transaction may fail. The user will have to try giving feedback again at a later time.
- The user's existing login may become invalid from inactivity. The user will then be prompted to re-enter their credentials.

**Concurrent Events**:

- Other users may be giving feedback to AIs at the same time.
- The AI will be processing existing questions in the database.

**State When Scenario Finishes**:

The web application has indicated that their feedback has been submitted and they can continue looking at their previously submitted questions. The database has feedback stored about the question for the AI to use in future processing.

## 2.07: User searches the global dashboard for a question

**Author**: Jackson Chandler
**Initial Assumptions**:

- The user is logged in to a valid session.
- There are questions in the database

**Normal Flow**:

The user logs in and navigates to the questions page there will be a search option to search for previously asked questions. It is set up in  a similar manner to the initial question creation page. Any questions that match the search criteria are displayed along with any data gathered for it.

**What Can Go Wrong**:

- The database transaction could fail, the user is notified

**Concurrent Events**:

- Other users may be searching for questions at the same time.
- The AI will be processing existing questions in the database.

**State When Scenario Finishes**:

A message will be displayed whether or any data for the question was found. If there is any data will also be displayed for the question.

## 2.08: Users sorts the questions on the dashboard

**Author**: Oliver Payne
**Initial Assumptions**:

- The database is up and running
- The user is logged in to a valid session
- There are questions in the database

**Normal Flow**:

The user navigates to the “Dashboard” page. From there, a complete list of questions in the learner database is displayed. Attributes of each question are displayed as headers in the columns of the table. Sorting arrows are shown on each label. The headers to the table columns are the question itself, status, and accuracy.

To sort the questions, the user clicks on the arrows on the headers. This sorts the questions in ascending or descending order of the particular header that was pressed.

**What Can Go Wrong**:

- The webpage could be out of sync with the database. The webpage then clears out the table and attempts to repopulate its entries by querying the database.
- User could unintentionally sort by a different criteria. The user then sorts the entries again based on his or her intended criteria.

**Concurrent Events**:

- Other users are adding new questions to the database and editing existing ones
- The status of the questions is being updated as the learners continue to process them

**State When Scenario Finishes**:

The complete listing of questions is sorted in the user view based on the sorting criteria that the user has selected.

## 2.09: User duplicates a question from the global dashboard without using existing AI models

**Author**: Cody Moffitt
**Initial Assumptions**:

- The database is up and running
- The user is logged in to a valid session
- A question exists for the user to duplicate

**Normal Flow**:

The user navigates to the global dashboard on the web application. The user chooses a question from another user that they wish to duplicate for their account. They click the duplicate button on the row for the question they wish to duplicate. A dialog box presents the options of using existing AI models, or creating new ones. The user chooses to create new ones.

**What Can Go Wrong**:

- The submission to the database may timeout. The user is then informed that the question was not submitted and that he or she should check with the system administrator if the problem persists.
- The user may submit a question that they have already posed. The user is informed when this happens.
- The user's login session may timeout if they take too long. The user is then prompted to renew the session by re-entering his or her credentials.

**Concurrent Events**:

- Other users may be entering/editing questions at the same time.
- The AI will be processing existing questions in the database.

**State When Scenario Finishes**:

The database will now have the duplicate question saved for this user and waiting for processing. The question will have a status of "Queued" until processed. The web application will indicate the question has been saved to the database for this user.

## 2.10: User duplicates a question from the global dashboard using previously created AI models

**Author**: Cody Moffitt
**Initial Assumptions**:

- The database is up and running
- The user is logged in to a valid session
- A question exists for the user to duplicate

**Normal Flow**:

The user navigates to the dashboard on the web application. The user chooses a question from another user that they wish to duplicate for their account. They click the duplicate button on the row for the question they wish to duplicate. A dialog box presents the options of using existing classifier models, or creating new ones. The user chooses to use the existing classifier models. If there are multiple instances of this question, the user will be shown a list of available AI models to duplicate. The list will indicate accuracy levels for each AI model, allowing the user to select which AI model to duplicate for their account.

**What Can Go Wrong**:

- The submission to the database may timeout. The user is then informed that the question was not submitted and that he or she should check with the system administrator if the problem persists.
- The user may submit a question that they have already posed. The user is informed when this happens.
- The user's login session may timeout if they take too long. The user is then prompted to renew the session by re-entering his or her credentials.

**Concurrent Events**:

- Other users may be entering/editing questions at the same time.
- The AI will be processing existing questions in the database.

**State When Scenario Finishes**:

The database will now have the duplicate question saved for this user and waiting for processing. The question will have a status of "Queued" until processed. The web application will indicate the question has been saved to the database for this user. The chosen AI models will be duplicated for this user for processing this question.

## 2.11: User asks similar question from global dashboard

**Author**: Jackson Chandler
**Initial Assumptions**:

- The database is up and running
- The user is logged in to a valid session
- There are similar questions in the database

**Normal Flow**:

The User navigates to the questions page of the dashboard. The user needs to choose a base question and additional options, then click search. Similar question include same base question with different options, or a different base question with with the same additional options.

**What Can Go Wrong**:

- User chooses the wrong additional options of the questions, they can delete the question or edit it.
- User chooses the wrong base question, they can delete the question or modify it.
- The database transaction fails, the user is notified.

**Concurrent Events:**

- Other users will be adding and deleting other questions during the search

**State When Scenario Finishes**:

When the search button is clicked, the results will be displayed if available. If not available then “No Matches” will be displayed.

## 2.12: Admin tells data-loader to force fetch new data

**Author**: Andrew Pachuilo
**Initial Assumptions**:

- The data mart has been created
- The database to fetch from is up and running and is in i2b2 format

**Normal Flow**:

Administrator opens up a terminal. They then navigate to the location where the data-loader source is running. They then execute a 'force-data-fetch' script that forces the data-loader to fetch new data to be used. The data loader then contacts the specified i2b2 database, and search for new data to add. It retrieves this new data and places it into the data mart.The previous data is removed. Once finished the user is notified that the operation was successfully completed. The data loader also generates a dated log detailing the fetch operation.

**What Can Go Wrong**:

- The connection to the database may be interrupted when downloading the new data. In this case the data mart will not change at all, and the error will be noted in the log file. The fetch will run again until it is successful or it fails for a user configurable amount of times.
- The data mart may have space constraints preventing the download of new data, while keeping the old data.This error will be noted in the log file, and the data mart will delete the old data and grab the new data.
- Also currently accessing the data mart may slow down or inhibit database access during the comparison operation. If the fetch fails, it will run again until it is successful or it fails for a user configurable amount of times.

**Concurrent Events**:

- Web application is accessing information in the data mart.
- AI is fetching data from the data mart for learning.
- Users will be adding/editing questions to the data mart

**State When Scenario Finishes**:

The data mart is now current with the target i2b2 database for the specified information.

## 2.13: Data loader runs fetch on schedule

**Author**: Cody Moffitt
**Initial Assumptions**:

- The data loader is configured
- The source database is available and running
- The data mart has been created

**Normal Flow**:

The data fetch command line utility runs with the specified configuration at the time it is set to run. It contacts the specified i2b2 database, and searches for newly added patient data. It retrieves this new data and places it into the local data mart. The data loader generates a dated log detailing the fetch operation.

**What Can Go Wrong**:

- The connection to the database may be interrupted when downloading the new data. In this case the data mart will not change at all, and the error will be noted in the log file. The fetch will run again until it is successful or it fails for a user configurable amount of times.
- The data mart may have space constraints preventing the download of new data, while keeping the old data.This error will be noted in the log file, and the data mart will delete the old data and grab the new data.
- Also currently accessing the data mart may slow down or inhibit database access during the comparison operation. If the fetch fails, it will run again until it is successful or it fails for a user configurable amount of times.

**Concurrent Events**:

- AIs will be accessing the data mart patient information for their operation.
- Users will be adding/editing/deleting questions from the data mart.

**State When Scenario Finishes**:

The data mart is now current with the target i2b2 database for the specified information. The data loader has logged the operation, either creating a new log file or appending to the existing data loader log.

**Notes**:

The new data will be constrained to what is desired for the data mart.

## 2.14: Admin edits data loader scheduler

**Author**: Jackson Chandler
**Initial Assumptions**:

- The data loader config file exists
- The data mart has been created

**Normal Flow**:

The admin starts a command line tool that they can use to edit the settings of the data loader. From there they pass parameters to be changed using regular expressions. The passed parameter is validated with the system to make sure all entries are correct. If correct the admin is notified of the change.

**What Can Go Wrong**:

- Parameters are not valid, the data loader will log the error and not take the new configuration.
- Parameters are not updated to the new settings and old parameters are still being used, the error will be logged.
- The admin may try to alter the schedule as the data loader is concurrently fetching data, the command line tool should reject the changes.

**Concurrent Events**:

- The data loader may be currently running

**State When Scenario Finishes**:

Once the data loader has been updated the admin will receive a message that it has done so. If new parameters are accepted, then they will be put into effect.

**Notes**

For example, the admin could alter the interval at which the data loader fetches data.

## 2.15: Admin changes i2b2 database

**Author**: Oliver Payne
**Initial Assumptions**:

- The data loader is already running with a different database

**Normal Flow**:

The user opens the config file with his or her choice of text editor. In the config file, there is a field for the source database that is currently being used. The user deletes the name of the database that has already been inserted and then inputs the desired database name. The user saves the config file and exits the editor.

The admin then initiates the changes to the data loader. He or she uses the corresponding command from the data loader tool. This informs the program that a change has been made to the config file. If data loading is in progress, the procedure will be completed before the change is made. Upon completion of the data loading in process, the data loader reads in the config file and then establishes a connection with the new database. If data loading is not in progress, the program immediately switches to the new database.

**What Can Go Wrong**:

- The user may input an invalid database name. The admin is informed of this when the program attempts to establish a connection to the new database. When the connection fails, the program continues to use the old database to load data.

**Concurrent Events**:

- The data loader is running, loading data from the database that it is already connected to
- The learners are continually processing the data that the loader has retrieved

**State When Scenario Finishes**:

The data loader is now connected to the new database. It resumes data loading by pulling from the new i2b2 database.

## 2.16: Admin creates data mart from script file

**Author**: Oliver Payne
**Initial Assumptions**:

- The user has the administrative privileges to run the script
- The source database is up and running

**Normal Flow**:

The user runs the script file, which creates the data mart. The user specifies the source database when running the script. When the data mart has been constructed, it attempts to establish a connection with the source database. Once the connection has been established, the user uses the fetch command on the data loader command line tool to manually load in data to the data mart.

**What Can Go Wrong**:

- The fetching procedure may fail, resulting in no data being entered into the data mart. The admin is informed that a problem has occurred.
- The user may run the script incorrectly, unintentionally creating the data mart with the wrong source database. If the database is an invalid name, the connection will fail and the admin will be informed.

**Concurrent Events**:

- The source database is already running, and other users are requesting data from it

**State When Scenario Finishes**:

The data mart is set up with the correct i2b2 standard. The data mart has also been loaded with data from the source database. Once configured, the data loader can now start pulling from the source database on schedule.

## 2.17: AI controller spawns instance of AI

**Author**: Andrew Pachuilo
**Initial Assumptions**:

- The AI controller and data mart are up and running.
- There is one question on the queue.

**Normal Flow**:

The AI controller detects a new question on the queue. The AI controller then uses that question with its given parameters and spawns an AI instance to process it. The AI instance uses an algorithm determined by the AI controller based on the question. It then notices the queue is now empty and starts trying to get better answers for previous questions with lower accuracies. It chooses an active question with a low accuracy and spawns an instance of the AI with an algorithm it finds to be the best fit for the question to test new parameters to optimize with. The AI continues the process of fetching questions and spawning instances of AIs until it reaches its specified cap of AI nodes.

**What Can Go Wrong**:

- The data mart is unavailable. The AI controller will continually ping the data mart until it is available. After a specified amount of time it will alert the administrator that something is wrong with the data mart.

**Concurrent Events**:

- AI instances are running
- People are adding/deleting/editing questions in the data mart

**State When Scenario Finishes**:

AI controller waits for new questions to reach get on the queue. Continues processing older questions until then.

**Notes:**

A best fit algorithm will be determined via continuous exploration and experimentation with algorithms used by questions with similar parameters.

## 2.18: AI processes a question

**Author**: Cody Moffitt
**Initial Assumptions**:

- Questions exist in the data mart for the AI to process.
- The data mart is accessible.
- The AI in question has been spawned.

**Normal Flow**:

The AI instance has finished processing a question. It marks the question status as “Need Feedback” in the data mart, and saves the answer provided by each of its classifiers for that question. The AI instance notifies the AI controller that it has finished processing. The AI instance saves its data to the data mart, and shuts down.

**What Can Go Wrong**:

- The data mart may be unavailable. The AI will continue to attempt to save its data until the data mart is back online and it is successful.
- The algorithms employed by the AI may fail or get caught in a loop, and it will never notify the AI controller that it has finished processing. This could cause lock ups if the instance goes rogue. The AI controller configuration should have a hard limit on how long an AI can take, allowing it to shut down rogue AI instances. The algorithms shouldn’t fail though, this a is a safety precaution.

**Concurrent Events**:

- Other AI instances will likely be processing questions.
- Users will be adding/creating/editing questions in the data mart.

**State When Scenario Finishes**:

The AI instance has been shut down and its model data has been saved, the question it was working on has been answered and marked as “Feedback Required” in the data mart.

## 2.19: Admin edits AI controller config file

**Author**: Andrew Pachuilo
**Initial Assumptions**:

- The AI configuration file exists
- Admin is authorized to change the config file.

**Normal Flow**:

Administrator navigates to the configuration file and opens it with their preferred editor. They can then choose to change any of the following settings: ‘Max AI Nodes’, ‘AI Timeout’, and ‘X algorithm disabled’. They scroll down to the section where they can disable/enable various algorithms. They scroll down to ‘NeuralNetwork_Disabled: False’ and change it to true. They then save the file to be applied to the AI controller later.

**What Can Go Wrong**:

- Administrator changes something they didn’t mean to or adds garbage to the config file. The administrator can fix their changes or use the starting config file.
- They rename the file and the AI controller can’t find the file for later use. The administrator will have to fix the name of the file.

**Concurrent Events**:

- AI controller is running.

**State When Scenario Finishes**:

The file is saved with a valid configuration state to be used later by the AI.

**Note**:

‘X algorithm’ would be a specified algorithm such as: neural network, k-trees, naive bayes, support vector machine, and random forest.

## 2.20: Admin applies AI config file to update AI controller

**Author**: Jackson Chandler
**Initial Assumptions**:

- Config file is valid

**Normal Flow**:

The administrator restarts the AI Controller. The new configuration is used by the AI Controller.

**What Can Go Wrong**:

- New config file is not being used, the admin should attempt to apply it again and observe the log for any errors.
- Improper format for config file, the error is logged and the new configuration is rejected.

**Concurrent Events**:

- AI Controller is running

**State When Scenario Finishes**:

The new config file is now in use, and the AI Controller is running again.

## 2.21: User is automatically logged out due to inactivity

**Author**: Cody Moffitt
**Initial Assumptions**:

- The data mart is up and running
- The user is logged in
- The AI config file is setup and valid

**Normal Flow**:

The user makes no web requests for a configurable specific amount of time. The system automatically invalidates their session.

**What Can Go Wrong**:

- The user intended to keep using the system, but was inactive. The login screen pops up after the user is logged out, so they may login again.

**Concurrent Events**:

- Other users are logging in and being logged out
- The data mart is being used by the AI and perhaps being updated

**State When Scenario Finishes**:

The user is logged out of the web application, a prompt is shown to login again to continue their session. The data mart has marked the user session as invalid.

# Appendix

## A.01: Notes/Issues/Assumptions

1. Access to the current KUMC i2b2 database may not be available for this project's duration. A simulation dataset may be provided.
2. The exact nature and scope of questions users may pose to the AI have not yet been determined.
    - Hospital readmittance chance is one of the questions, if not the only one.
3. Implementation choices (programming languages, web server framework, …) have not been determined by KUMC and are up to the student NEMO team.
4. An exact list of required classifier and optimization algorithms to implement has not yet been determined, but it was noted that a modular framework was wanted to easily add and drop new algorithms.
5. Server availability and architecture is not yet determined.
6. May need to implement different user types, with different privileges, but this hasn't been determined fully yet.
7. Possible limit needs to be created for the number of questions a user may ask.
8. The database that the user creates using the script file will be used to contain both test data and learner data for the AIs. The data mart is constructed to receive i2b2 formatted entries. All entries will conform to this standard.