CodeTrix

Attack Simulator on Recommender Systems

INB302 - Capstone Project

John Wilkinson, Philippe Gref-Viau, Corey Thompson, Mikhail Sutormin

Supervisor: Associate Professor Yue Xu

2014

Semester 1

CONTENTS

SECTION 1 PROJECT SUMMARY

INTRODUCTION

This document will serve as a guide describing in detail the features and characteristics of the product built by our team, CodeTrix, in the context of this semester's INB382 Capstone Project. This product primarily consists of a digital platform used to facilitate the testing of known attacks upon existing recommender systems.

Recommender Systems are used widely by companies such as Amazon, EBay, Spotify and many other companies to give recommendations to their users.

The Attack Simulator on Recommender Systems project supports important research into collaborative filtering recommender systems used widely online. By pursuing this area of research, Australia can lead the way in providing systems that are effective in producing recommendations relevant to the user, and robust enough to avoid manipulation.

Recommendations in the past have been from person to person, and the characteristics of the person could be taken into account when deciding whether to act or not on their recommendation. Collaborative filtering on the other hand, massively increases the data used to arrive at a recommendation, which provides huge benefits in versatility, accuracy and overall usability, however, the use of mass amounts of data makes it vulnerable to manipulation due to the fact that users are able to remain anonymous. Attackers, who cannot be readily distinguished from real users can inject fake profiles into the system to distort the results and in turn, produce recommendations which can unfairly produce better results for the attacker. Attackers can also use public accessible data to enhance the precision, invisibility and effectiveness of their attacks.

In the Attack Simulator on Recommender Systems project, we build an application to simulate these attacks on recommender systems. The resulting data after the simulated attack can then be compared with the original data to illustrate the effectiveness of different types of attacks on different types of recommender systems.

With the power of collaborative filtering too great to ignore, and vulnerabilities that threaten to negate or abuse this power, facilitating the research into this field is paramount.

REQUIREMENTS

FUNCTIONAL REQUIREMENTS

1. Select datasets

The user will be able to select one or more datasets, containing the relevant information (user-item-rating groupings), to be analyzed by the application

Precondition	*None*
Path	1. The user starts up the application 2. The system provides the necessary fields (based on **1.3** & **1.4**) 3. The user fills out the necessary fields
Alternative Path	1. The user starts up the application 2. The system provides the necessary database fields (based on **1.3 & 1.4**) 3. The user fills out the necessary database fields
Exceptions	
Postcondition	All the relevant fields are filled out and contain valid input

1.1. Select entry dataset

The user will be able to select an unaltered dataset to serve as a pre-attack basis sample

Precondition	*None*
Path	1. The user starts up the application 2. The system provides the necessary fields (based on **1.4**)
Alternative Path	*None*
Exceptions	
Postcondition	Must respect **1.1.1** and **1.1.2**'s postconditions.

1.1.1. Select entry dataset from text file

The user will be able to extract the sample entry dataset's data from a formatted text file

Precondition	The data source must be set to the "text file" option (see **1.4**)
Path	1. The user starts up the application 2. The system provides the necessary fields (based on **1.4**) 3. The user fills out the dataset file field
Alternative Path	*None*
Exceptions	• **Invalid file type**: The selected file does not correspond to a .txt file • **Invalid file path**: The provided file location does not match an existing file path • **Access violation**: The file at the provided location cannot be accessed by the file system
Postcondition	The entry dataset file location field must contain a valid file path

1.1.2. Select entry dataset from database

The user will be able to extract the sample entry dataset's data from a specific database

Precondition	The data source must be set to the "database" option (see **1.4**)

Path	1. The user starts up the application 2. The system provides the necessary fields (based on **1.4**) 3. The user fills out the database attribute fields
Alternative Path	*None*
Exceptions	*None*
Postcondition	All the relevant database attribute fields must be filled out and valid

1.1.2.1. Provide database hostname/IP address

Precondition	The data source must be set to the "database" option (see **1.4**)
Path	1. The user starts up the application 2. The system provides the necessary fields (based on **1.4**) 3. The user fills out the database hostname/IP attribute field
Alternative Path	*None*
Exceptions	**Missing value**: provided field value is null or empty
Postcondition	The hostname/IP field contains a valid value

1.1.2.2. Provide database username

Precondition	The data source must be set to the "database" option (see **1.4**)
Path	1. The user starts up the application 2. The system provides the necessary fields (based on **1.4**) 3. The user fills out the database username field
Alternative Path	*None*
Exceptions	**Missing value**: provided field value is null or empty
Postcondition	The username field contains a valid value

1.1.2.3. Provide database password

Precondition	The data source must be set to the "database" option (see **1.4**)
Path	1. The user starts up the application 2. The system provides the necessary fields (based on **1.4**) 3. The user fills out the database password attribute field
Alternative Path	*None*
Exceptions	**Missing value**: provided field value is null or empty
Postcondition	The password contains a valid value

1.1.2.4. Provide database port number

Precondition	The data source must be set to the "database" option (see **1.4**)
Path	1. The user starts up the application 2. The system provides the necessary fields (based on **1.4**) 3. The user fills out the database port number attribute field
Alternative Path	*None*
Exceptions	**Missing value**: provided field value is null or empty **Integer parsing exception**: the field value cannot be parsed to an integer
Postcondition	The port number field contains a valid value

1.1.2.5. Provide database schema containing the dataset data

Precondition	The data source must be set to the "database" option (see **1.4**)
Path	1. The user starts up the application 2. The system provides the necessary fields (based on **1.4**) 3. The user fills out the database password attribute field
Alternative Path	*None*
Exceptions	**Missing value**: provided field value is null or empty
Postcondition	The password contains a valid value

1.2. Select pre-computed attack dataset

The user will be able to select a pre-computed set of groupings designed to simulate an attack on the sample dataset.

1.2.1. Select pre-computed attack dataset from text file

The user will be able to extract the "attack" dataset's data from a formatted text file

1.2.2. Select pre-computed attack dataset from database

The user will be able to extract the "attack" dataset's data from a specific database

1.3. Choose if attack dataset should be generated manually or not

The user will be able to choose whether the attack set will be provided as a pre-computed dataset (1.2) or will be generated through the application (6)

Precondition	*None*
Path	1. The user starts up the application 2. The user selects if the dataset should be generated manually or not (default is "yes")
Alternative Path	*None*
Exceptions	*None*
Postcondition	A choice concerning the dataset generation has been set

1.4. Choose the type of the dataset's source

The user will be able to choose whether the datasets to download come from text files or from a database.

Precondition	*None*
Path	1. The user starts up the application 2. The user selects the source of the datasets (text file or database)
Alternative Path	*None*
Exceptions	*None*
Postcondition	A choice concerning the datasets' source has been set

2. Download datasets

The user will be able to import the data from the provided datasets into the application

Precondition	All the necessary fields described in **1** have been filled out
Path	*(continued from **1**)* 1. Trigger the download

Alternative Path	*None*
Exceptions	**Database connection exception**: cannot connect to the database with the provided information **File exception**: cannot open or access one of the system files
Postcondition	All the data contained in the selected datasets has been downloaded into the application correctly

3. Save current database info

The user will be able to save the current database credentials so they can be easily

Precondition	The database info fields are filled in, and the dataset source is set to "database"
Path	1. The user starts up the application 2. The system provides the necessary database fields (based on **1.3 & 1.4**) 3. The user fills out the necessary database fields 4. The user saves the database info as a reusable definition file
Alternative Path	*None*
Exceptions	**Invalid file name:** the selected file name is invalid **Access violation:** the path of the file's location is inaccessible **Missing information:** the value for some of the fields is missing
Postcondition	The current database credentials are persisted in a named file

4. Load existing database info

The user will be able to fill out the database connexion info fields by loading a pre existing configuration from a file.

Precondition	The dataset source is set to "database"
Path	1. The user starts up the application 2. The system provides the necessary database fields (based on **1.3 & 1.4**) 3. The user selects a file to get the information from
Alternative Path	*None*
Exceptions	**Access violation:** the path representing the file's location is inaccessible **Format exception:** the formatting of the file makes the reading of the parameters impossible
Postcondition	The database fields are filled with the values persisted within the selected file

5. Dataset generation page navigation

The user will be able to navigate to the dataset generation page

Precondition	- The data from the entry dataset is correctly loaded into the application - The user triggered a move to the next page - The option to generate the attack dataset manually is selected - The previous page is the dataset selection page
Path	*(continued from **2**)* 1. Hit the "next page" button
Alternative Path	*None*
Exceptions	*None*

Postcondition	The application displays the dataset generation page

6. Generate attack dataset

The user will be able to generate a dataset aimed at affecting the results of the recommender system algorithm's computations

Precondition	- The data from the entry dataset is correctly loaded into the application - The option to generate the attack dataset manually is selected
Path	*(continued from **5**)* 1. Go through steps **6.1**, **6.2**, and **6.3**
Alternative Path	*None*
Exceptions	*None*
Postcondition	An attack dataset has been generated by the application

6.1. Choose items to promote

The user will be able to choose what items will try to be promoted through the attack

Precondition	*None*
Path	1. Select the items to promote in a list of all available items
Alternative Path	1. Choose a pre-determined number of items to select 2. Let the application choose that same amount of items at random
Exceptions	*None*
Postcondition	A certain number of items to be promoted have been selected from the pool of available items

6.2. Choose number of fake profiles to create

The user will be able to select the total number of items that will be generated for the attack

Precondition	*None*
Path	1. Input the number of items to be generated
Alternative Path	*None*
Exceptions	**Out of range:** the number of items to create is not within the range of allowed values **Missing value:** no value was provided for this parameter
Postcondition	A number of items to be generated has been set

6.3. Select attack filling type

The user will be able to select the type of filling method (randomized or targeted) to use while generating the attack elements

Precondition	*None*
Path	1. Select between the two filling methods
Alternative Path	*None*
Exceptions	*None*
Postcondition	One of the filling methods has been chosen

7. Save attack dataset

The user will be able to save the generated attack dataset as a named file for subsequent use

Precondition	An attack dataset must have been generated
Path	1. Generate the attack dataset 2. Save the dataset as a named file
Alternative Path	*None*
Exceptions	**Invalid file name:** the selected file name is invalid **Access violation:** the path of the file's location is inaccessible
Postcondition	The generated dataset is persisted in a file

7.1. Save attack dataset to text file

7.2. Save attack dataset to database

8. User selection page navigation

The user will be able to navigate to the user selection page

Precondition	- The data from the entry dataset is correctly loaded into the application - The data from the attack dataset is correctly loaded into the application - The user triggered a move to the next page
Path	1. Select datasets (see **1**) with manual dataset generation turned off 2. Hit the next page button
Alternative Path	1. Select datasets (see **1**) with manual dataset generation turned on 2. Hit the next page button 3. Generate the attack dataset (see **6**) 4. Hit next page button
Exceptions	*None*
Postcondition	The application displays the user selection page

9. Select user

The user will be able to select which user's recommendations will be affected by the attack

Precondition	- The data from the entry dataset is correctly loaded into the application - The data from the attack dataset is correctly loaded into the application
Path	(*continued from **8***) 1. The user selects a user in a list of all available users
Alternative Path	*None*
Exceptions	*None*
Postcondition	A sample user has been selected

10. Items selection page navigation

The user will be able to navigate to the items selection page

Precondition	- The data from the entry dataset is correctly loaded into the

	application - The data from the attack dataset is correctly loaded into the application - The user triggered a move to the next page - A sample user has been selected
Path	1. Select user (see **9**) 2. Hit the next page button
Alternative Path	*None*
Exceptions	*None*
Postcondition	The application displays the items selection page

11. <u>Select items to compute ratings predictions for</u>

The user will be able to choose a subset among the available items which predictions' will be displayed

Precondition	- The data from the entry dataset is correctly loaded into the application - The data from the attack dataset is correctly loaded into the application - A sample user has been selected
Path	*(continued from **10**)* 1. Select the items whose ratings will be computed out of a list
Alternative Path	*(continued from **10**)* 1. Select the items whose ratings will be computed out of a list 2. Select items out of a subgroup representing the promoted items
Exceptions	*None*
Postcondition	- The data from the entry dataset is correctly loaded into the application - The data from the attack dataset is correctly loaded into the application - A sample user has been selected - An item subset has been chosen

12. <u>Results page navigation</u>

Precondition	- The data from the entry dataset is correctly loaded into the application - The data from the attack dataset is correctly loaded into the application - A sample user has been selected - An item subset has been chosen - The user triggered a move to the next page
Path	3. Select user (see **9**) 4. Hit the next page button
Alternative Path	*None*
Exceptions	*None*
Postcondition	The application displays the items selection page

13. <u>Analyze attack results</u>

NON-FUNCTIONAL REQUIREMENTS

FILE FORMAT STANDARDS

The application will have to be able to read out of the provided dataset file format. The dataset format that will be adopted for this experiment will be based on the GroupLens Research Project's anonymous collection of ratings. Each entry in the file is organized in the following way:

<userId> <itemId> <rating> <ratingDate>

UserId and itemId are both unique identifiers respectively representing individuals and the movie that has been rated. The rating will be an integer score between 1 and 5. The rating date serves no current purpose, and will be ignored by the application. The rest of the parameters will have to be loaded up correctly within the application, which will in turn be able to generate data in the same format when exporting datasets for future usage.

NON-MANUAL PERSISTENCE

In the spirit of usability, some fields should retain their previous values after each experiment, even after the application has shut down. For example, the last used dataset file path should be automatically input by the application in order to potentially avoid locating the file for each experiment. The same goes with the database connexion parameters.

EXTENSIBILITY AND SCALABILITY

The system will have to be designed to accommodate future changes by other teams of students, as the product will evolve over the coming semesters after CodeTrix has completed its initial instantiation. This implies a flexible and modular approach to the application's architecture, documentation and ideally pre determined unit tests to ensure the quality of later versions.

PERFORMANCE

The application will be running work intensive algorithms on large datasets. In this instance, the computation time can tend to increase dramatically. That being said, great attention will have to be given to the overall performance and speed of the computations. The team's focus will be set on improving the selected recommender algorithm's efficiency to its fullest potential as well as ensuring reasonable computation times in the order of seconds, not minutes, in order to make the application viable as an analysis platform.

SCOPE

The final product will be used in a research context by our main supervisor, Pf Yue Xu, of the Queensland University of Technology's Computer Science department. In this sense, it will be designed in close participation with Pf Xu and will attempt to reflect her target needs while being flexible enough to be modified subsequently by future teams of students.

The actual expected scope of this project by the time of its completion will include a functional, professional and streamlined graphical user interface (GUI) that will allow researchers to generate predictions based on specified data sets, using recommender systems, and subsequently easily launch attacks against these data sets, while collecting and formatting any relevant generated results. More specifically, it will allow the user to set the different parameters defining the nature of the attacks, choose the datasets from various sources, such as a text file or a database and save the results of the experiments to those same sources.

According to the nature of the attacks, some of the final product's functionalities will also focus on generating fake user profiles, each with one or more rating to contribute to a particular attack. This generation process will also involve choosing certain items of interest to "promote" and tracking those items throughout the process.

QUALITY OBJECTIVES

The application will provide accurate and usable results which will be ensured through thorough testing during application development.

The application can be used for demonstrations, so the GUI will be user-friendly, consistent and provide information in an organised and efficient way.

The application will be stable and capable of processing large amounts of dataS

SECTION 2 COMPARISON OF PLANNED VERSUS ACTUAL

HISTORY OF APPROVED CHANGES

Item	Original	Approved Changes	Comments
Application language	Java	C#	Difficulty implementing GUI with Java, so switched to C# due to prior experience

ORIGINAL SCHEDULED DEADLINE VS. ACTUAL COMPLETION DATE

Item	Schedule	Completion
Team Contract	11/03/2014	11/03/2014
Project Plan	25/03/2014	24/03/2014
Team Conflict Presentation	02/04/2014	02/04/2014
Finished Application	28/05/2014	
Final Report	21/05/2014	30/05/2014
Project Presentation	28/05/2014	28/05/2014
Showcase Poster	04/06/2014	
Project Showcase	04/06/2014	
Slides for Presentation	26/05/2014	26/05/2014

TEST PLANS AND TEST RESULTS

To check the correctness of our prediction algorithm we have implemented it in Excel.

TEST CASES

Our test consists of 5 users rating 32 movies. Based on their similarity we predict a rating of 33rd movie for target user. Below are results from Excel spreadsheet.

Item\User	Bob	Mark	Jill	Jake	Mary
Movie 1	3	4	3	3	3
Movie 2	4	4	4	4	4
Movie 3	2	2	2	2	2
Movie 4	1	1	1	2	1
Movie 5	4	4	5	4	4
Movie 6	2	2	3	2	2
Movie 7	2	3	2	2	2
Movie 8	3	3	3	3	3
Movie 9	2	2	2	2	3
Movie 10	2	2	2	2	2
Movie 11	3	2	3	3	3
Movie 12	2	2	2	2	2
Movie 13	4	4	4	4	4
Movie 14	3	3	3	3	3
Movie 15	3	3	3	3	3
Movie 16	3	3	3	4	3
Movie 17	2	2	2	3	2
Movie 18	1	1	1	2	1
Movie 19	5	5	5	5	5
Movie 20	3	3	3	3	3
Movie 21	2	2	3	2	2
Movie 22	3	3	4	3	3
Movie 23	3	3	2	3	2
Movie 24	2	2	2	1	2
Movie 25	3	4	3	3	3
Movie 26	2	2	3	2	3
Movie 27	2	2	2	2	2
Movie 28	1	2	1	1	1
Movie 29	2	2	2	2	1
Movie 30	4	4	4	4	4
Movie 31	2	2	2	2	3
Movie 32	2	2	2	2	2
Movie 33		3	2	3	2
Average	**2.56250**	**2.65625**	**2.68750**	**2.65625**	**2.59375**

Similarity	
Bob&Mark	**0.9176050**
Bob&Jill	**0.9125611**
Bob&Jake	**0.9144154**
Bob&Mary	**0.9141962**
Prediction for Bob	**2.65073**

SECTION 3 OUTSTANDING ISSUES

With further investment, the project can be expanded to facilitate the simulation of a wider range of attack and filtering algorithms which can be used as a catalyst for increased research speed and results, and therefore put Australia at the forefront in its use of Recommendation Systems.

Looking forward, we have a clear trajectory for the Attack Simulator on Recommender Systems project into the future.

- Enhance the viewing options when comparing results with multiple sets of pre-attack and post-attack data.
- The precision of attacks can be enhanced and automated, to better facilitate the simulation of varying attack types such as bandwagon, reverse-bandwagon and segment attacks.
- The collaborative filtering options can be expanded from just user-based collaborative filtering to include item-based and semantically enhanced collaborative filtering
- The addition of extra evaluation metrics and displays, which can also be exported for further use outside the application.

APPENDIX PROJECT DOCUMENTATION LIST

TECHNICAL REPORT

DATABASE INFORMATION

The project can import information into a new database from a plain text document that fits the schema of userID, itemID, rating. Alternatively, an existing database can be used that has the same schema.

The database consists of one table with the following fields

- userId
- itemId
- rating

CLASSES

The following classes contained in the "*Utility Folder*" are used to inject fake profiles into the database

- *FakeProfilesGenerator* class

METHODS

- `public void generateFakeProfiles(int numOfFakeProfilesToCreate, List<Item> promotedItems, bool isUserRatingFillingRandom)`
- `private long generatedUnusedUserId()`
- `private List<TableEntry> getTargetedRatingFillingsForUser(User fakeProfileUser, List<Item> unPromotedItems )`
- `private List<TableEntry> getRandomRatingFillingsForUser(User fakeProfileUser, List<Item> unPromotedItems)`

ALGORITHM

USER-BASED COLLABORATIVE FILTERING

The user-based collaborative filtering method (*k*NN) is as follows;

1. Calculate similarity of all users to target user using correlation formula below

   u = target user
   v = neighbouring user
   I = set of all items that can be rated
   $r_{u,i}$ = rating of item *i* for target user *u*
   $r_{v,i}$ = rating of item *i* for neighbouring user *v*
   $\bar{r}_u$ = average of ratings *u* over those items in I that *u* and *v* have in common
   $\bar{r}_v$ = average of ratings *v* over those items in I that *u* and *v* have in common

$$sim_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u) * (r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} * \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}}$$

2. Select *k* most similar users to the target user
3. Filter out any users with similarity rating of less than 0.1
4. Calculate prediction of item *i* for target user *u* using formula below

   V = set of *k* similar users
   $\bar{r}_u$ = average rating for target user
   $r_{v,i}$ = rating of item *i* for neighbouring user *v*
   $\bar{r}_v$ = average rating for neighboring user over all rated items
   $sim_{u,v}$ = mean-adjusted Pearson correlation from Step 1

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in V} sim_{u,v} * (r_{v,i} - \bar{r}_v)}{\sum_{v \in V} \left| sim_{u,v} \right|}$$

Note: Formula 1 calculates Pearson’s correlation coefficient. It has several requirements that need to be taken into consideration.

First, population e.g. set of ratings must have normal distribution. To check for “normality” population must have at least 30 values. It sets a minimum amount of ratings per user at 30 before any similarity with other users can be calculated.

Second, population cannot have critical values, that is values which are very different from the majority. In our case it’s true.

Other things to consider. The nature of population of ratings is that values are significantly different from each other compared to the whole range. Any change in rating by 1 point has a great effect on correlation coefficient. To qualify for 0.1 similarity cut off rate ratings between users must be at least 90% identical. This can create a situation where having tens of users with tens of ratings, none of them would qualify the criteria. Temporary solution could be changing the cut off rate to 0.2 which would still return acceptable results.

Another characteristic of Pearson’s correlation coefficient formula with given population is its numerical instability with certain patterns of ratings. See table below for example.

Item\User	Bob	Mark
Movie 1	4	2
Movie 2	5	3
Movie 3	4	2
Movie 4	5	3
Movie 5	4	2
Movie 6	5	3
Movie 7	4	2
Movie 8	5	3
Movie 9	4	2
Movie 10	5	3
Movie 11	4	2
Movie 12	5	3
Movie 13	4	2
Movie 14	5	3
Movie 15	4	2
Movie 16	5	3
Movie 17	4	2
Movie 18	5	3
Movie 19	4	2
Movie 20	5	3
Movie 21	4	2
Movie 22	5	3
Movie 23	4	2
Movie 24	5	3
Movie 25	4	2
Movie 26	5	3
Movie 27	4	2
Movie 28	5	3
Movie 29	4	2
Movie 30	5	3
Movie 31	4	2
Movie 32	5	3
Average	**4.50000**	**2.50000**

Similarity
1.0000000

Situation like that is almost impossible to occur under normal circumstances however can present an vulnerability for a possible attack.

TEAM PROCESS

CodeTrix implemented an Agile methodology for the project, holding weekly scrum meetings.

Date: 05/03/2014	Tasks
Key Decisions	Group to Finish Team Contract Communications through Facebook, Skype, and Email Group to research Project Options
John	
Phil	
Corey	First draft of Team Contract
Mikhail	

Date: 12/03/2014	Tasks
Key Decisions	We need to meet project supervisors Project Plan needs to be finalised
John	Research Project Options
Phil	Meeting with Supervisor
Corey	First draft of Project Plan, Meeting with Supervisor
Mikhail	Research Project Options

Date: 19/03/2014	Tasks
Key Decisions	We need to finalise project choice
John	Research Project Options
Phil	Meeting with project supervisor
Corey	Submit Project plan as possible, though we have no project yet, Meeting with Project Supervisor
Mikhail	Research Project Options

Date: 26/03/2014	Tasks
Key Decisions	John will do presentation of Team Conflict Scenario
John	Prepare Team Conflict arguments
Phil	Prepare Team Conflict arguments
Corey	Prepare Team Conflict arguments
Mikhail	Prepare Team Conflict arguments

Date: 02/04/2014	Tasks
Key Decisions	John and Mikhail responsible for documentation Phil and Corey responsible for Application

	development
John	Create Gui for Database Selection
Phil	Importing of database info
Corey	Database design and setup
Mikhail	Setup code behind for GUI for database selection

Date: 09/04/2014	**Tasks**
Key Decisions	More time effective to keep Corey and Phil on programming, and Mikhail and John on Documentation
John	Begin Final Document
Phil	Finish GUI for importing of ratings, work on algorithms
Corey	Setup Hibernate with database and application
Mikhail	

Date: 16/04/2014	**Tasks**
Key Decisions	
John	Project Final Documentation progress
Phil	Debug algorithms
Corey	Create slides for Project Review presentation
Mikhail	Begin work on Test Cases with Excel

Date: 30/04/2014	**Tasks**
Key Decisions	
John	Final Document Progress
Phil	Add more commenting to code
Corey	
Mikhail	Finish excel sheet for test cases

Date: 07/05/2014	**Tasks**
Key Decisions	Application to be re-built in C#
John	Final documentation progress
Phil	Build application from scratch in C#

Corey	Setup database integration with new project application
Mikhail	Research possible algorithm solutions

Date: 14/05/2014	**Tasks**
Key Decisions	
John	Submit Project Summary, progress on final report
Phil	Implement Algorithm modifications
Corey	Finish GUIs
Mikhail	Make test cases and add to final report. Amend algorithm section in final report with formula modifications

Date: 21/05/2014	**Tasks**
Key Decisions	John to do Final Presentation
John	Progress on Final Report, Finish Presentation Slides
Phil	Finish Application and create scenario for presentation
Corey	Finish Application and create scenario for presentation
Mikhail	Finish test cases and final report modifications

Date: 28/05/2014	**Tasks**
Key Decisions	
John	Finish Final Report, Finish Showcase poster
Phil	Work on scenario for Showcase, review final report
Corey	Work on scenario for showcase, review final report
Mikhail	Finish test cases and final report modifications