DECLARATION

I certify that:

1) No library has been used except bootstrap.js and jQuery.js. No machine learning library or graphics related library is used and all codes are original work by me.
2) This work is in compliance to the assignment undertaken for the recruitment process of Lrnr.
3) I have given due credit to them by citing them in this report and giving their details in the references.

Date: 7 March, 2017

Place: Kharagpur

Shyam Swaroop

Final Year UG Student
IIT Kharagpur

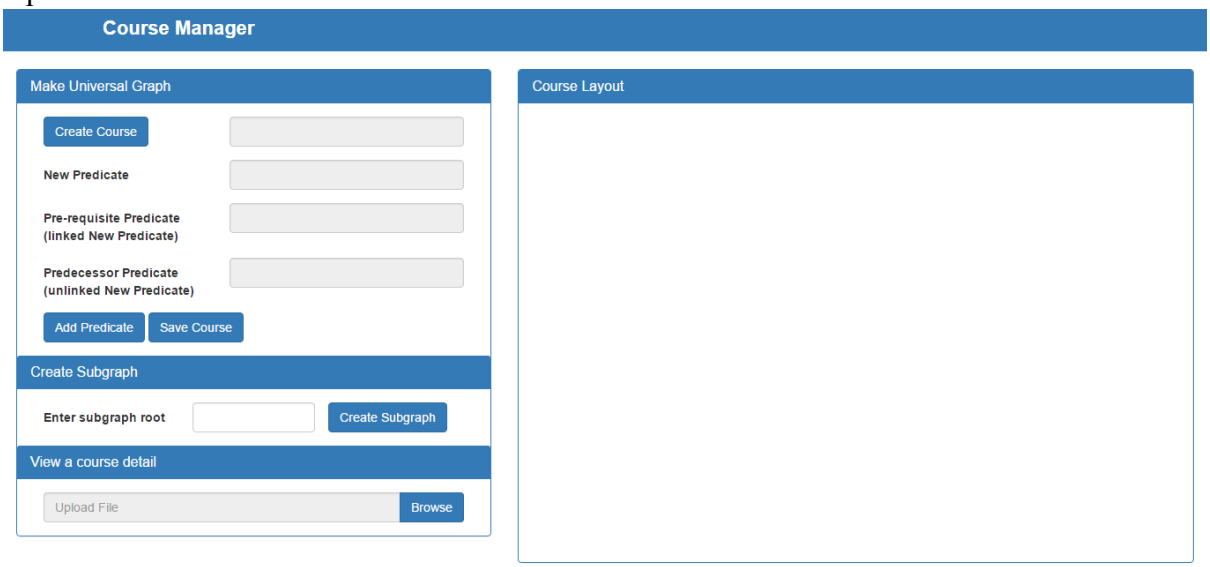
Part One: First part of the assignment says:

Write code to:
1) Define the Universal Graph shown above. (code should be able to take any universal graph)
2a) Enable selection of sub-graph as 'syllabus' (User Input)
OR
2b) Define sub-graph as 'syllabus'

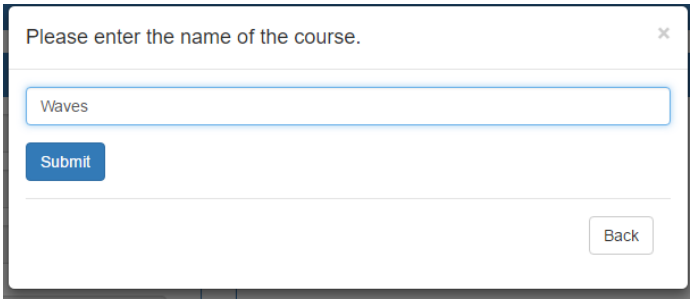
---

1) For this, a web page has been created, which can perform following operations:
   1. Create a course.
   2. Save it in JSON format.
   3. Saved course can be later accessed/viewed/edited (currently only further addition of predicate is allowed). A decoder to convert saved file into graph again is included.
   4. Sub-graph of created course or saved courses can be accessed/viewed/edited (currently only further addition of predicate is allowed).

2) Create a course and save it:
   1. Open "**finalUX.html**". Initial screen will look as follows:

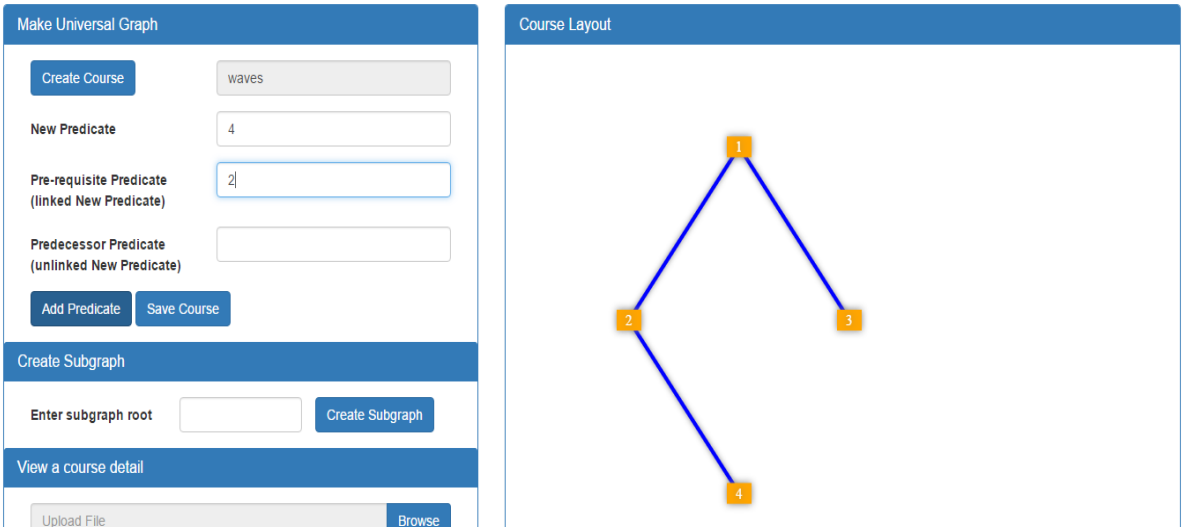


   2. Click on "**Create Course**" button. A dialog box as shown below will appear. Write the course name and click on "**Submit**" button. (Please, do NOT press Enter key, sorry for the bug.)

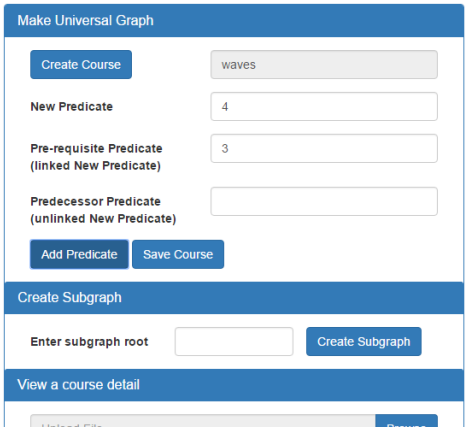


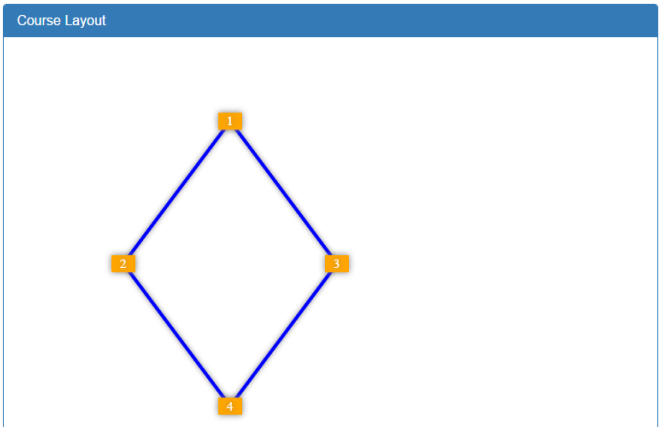
3. Here new predicate can be created in three ways:
   i) A new predicate with **NO pre-requisite predicate** - Enter "**New Predicate**" or topic-id and click on "**Add Predicate**" button.
   ii) A new predicate with **pre-requisite predicate and linked in the graph** - Enter "**New Predicate**" or topic-id, enter "**Pre-requisite Predicate**" and click on "**Add Predicate**" button.
   iii) A new predicate with **pre-requisite predicate and NOT linked in the graph** - Enter "**New Predicate**" or topic-id, enter "**Predecessor Predicate**" and click on "**Add Predicate**" button.
4. You can view the corresponding course structure/graph in the "**Course Layout**" section while creating and also later view the course details.

5. In cases when two or more pre-requisite concepts are present, you need to link them one by one. For example, consider you want to link predicate 4 to predicate 2, do it as shown below:



Now you want to link 3 to 4 also, again insert 4 in "**New Predicate**" and 3 in "**Pre-requisite predicate**" as shown below:

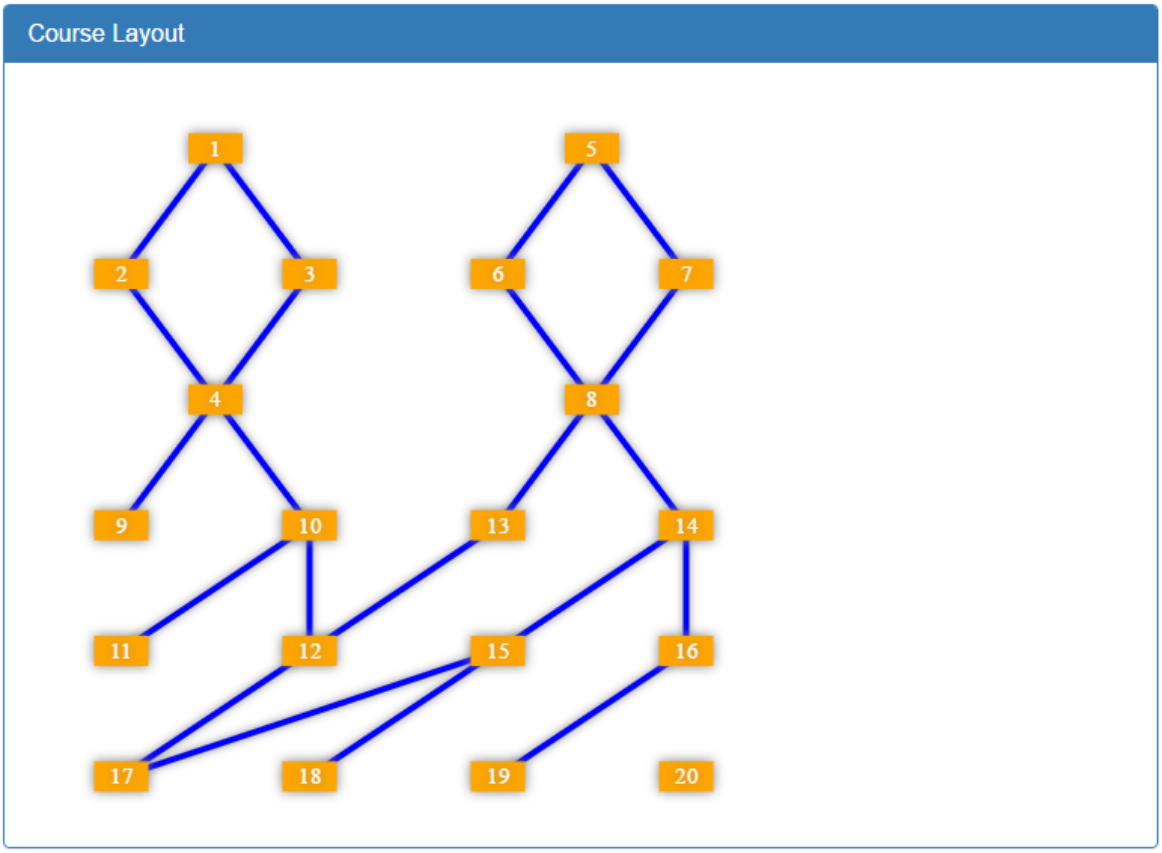




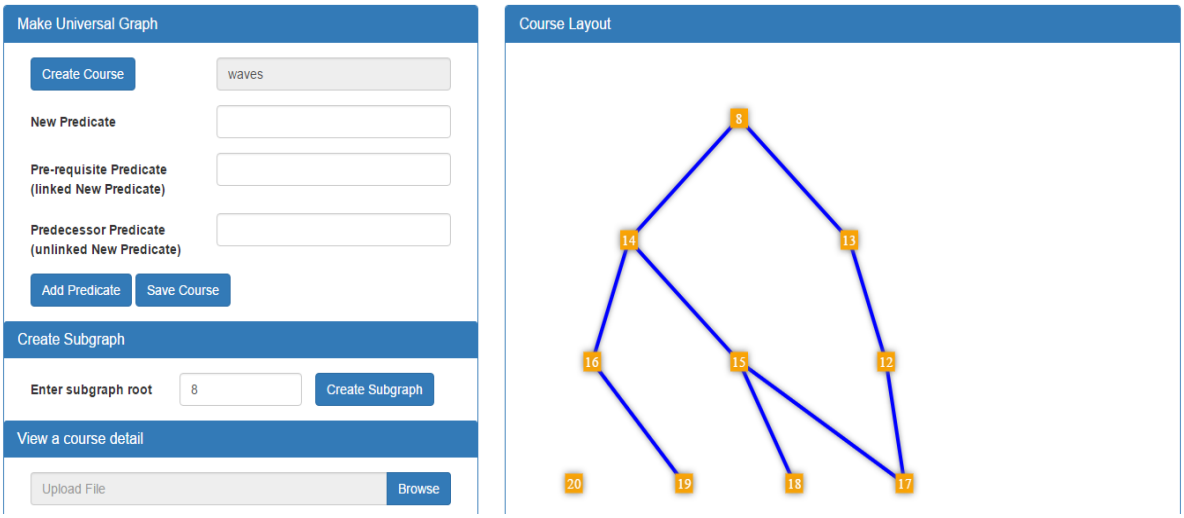
NOTE:

- The graph will always hold the properties of Directed Acyclic Graph (DAG).
- So, even when changes are made in one vertex of graph, this change will percolate to all the nodes in the vertex's sub graph.
- So, linking more pre-requisite predicates or child predicates to a vertex can be done in any sequence. Removing a predicate once it has been attached to the graph is not enabled due to lack of information on the purpose of this assignment.
- By convention, **all edges always point downwards**.
- Sorry, for poor graphics.

An example of course created:



6. When done adding predicates, click on "**Save Course**" button. A JSON file will be downloaded on your computer. No server based action will be performed in this entire assignment.
7. **This JSON file will be needed later when using Recommender System or Viewing course details later.**
8. A sub graph can be accessed by providing it's beginning node-id in "**Enter subgraph root**" and click "**Create Subgraph**". For example for the course shown in above figure, if we want to draw sub graph with root at 8:



END OF PART ONE OF ASSIGNMENT

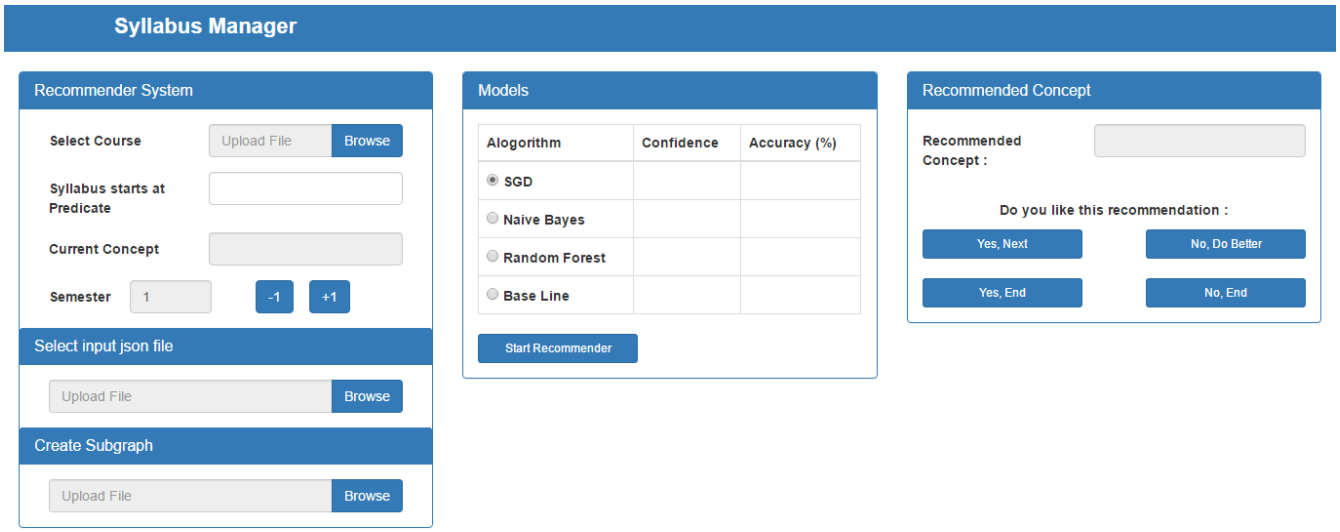
Part Two: Challenge 1 says:

Write a simple recommender system:
Set configuration:
consider_predicate_not_part_of_syllabus_depth=1 (if its 0, predicate nodes that are not part of syllabus will not be recommended)
success_criteria=0.7 (70% of nodes that are traversed should be successful)
i) Prompt beginning node of 'syllabus' (use order number to determine the starting point)
ii) User can input 0 or 1. Recommend next node based on Input. (0-fail, 1-pass)
iii) Code should recommend predicate nodes that are not part of syllabus based on configuration setting
above.
iv) Once 'success_criteria' is reached, show a message "successful" along with next recommendation.
v) Once all nodes are set to "1", show a message "You aced it" and exit the program.
vi) Have a hot "key word" to exit the recommender at any time.

Three online-learning Recommender System has been employed. These are classification based recommender systems, as such systems are reported to perform very well in case of online-learning recommendation systems.

3) Following assumptions were taken due to lack of information on this assignment:
   1. A predicate can only be a valid recommendation candidate when all it's pre-requisite predicates has been completed by the user.
   2. Purpose of semester was open-ended so nothing has been done regarding this.

4) How to use Recommender System?
   1. Open "**finalCS.html**". Initial screen looks like this:



2. "**Select Course**" by clicking browse. Now you will need the JSON files created by using PART One of this assignment. Select the JSON file after clicking on browse.
3. Give the beginning node of the syllabus in "**Syllabus starts at predicate**".
4. Choose an algorithm, and click on "**Start Recommendation**".
5. Recommendation will appear on rightmost "**Recommended Predicate**" section.
6. Click on "**Yes, Next**" if you like the recommendation and wish to continue getting recommendations.
   Click on "**No, Better**" if you don't like the recommendation and you want to get better recommendation.
   Click on bottom two buttons if you wish to end getting recommendations.
7. Accuracy of recommender system can be seen alongside.

NOTE:

1. To predict confidence (similar to classification margin), proper metric is needed as mentioned by Breiman, due to lack of proper information, it has been left blank as of now.
2. Base Line is deploying just uniform sampling technique.
3. **Some already created JSON files are provided in the shyam_lrnr_assignment directory. To view these course you can use, "finalUX.html" as mentioned in Part One.**

Part Two: Challenge 2 says:

Write a simple recommender system:
Set configuration
consider_predicate_not_part_of_syllabus_depth=1 (if its 0, predicate nodes that are not part of syllabus
will not be recommended)
success_criteria=0.7 (70% of nodes that are traversed should be successful)
i) accept JSON array [node_id,…] and convert to "syllabus" subgraph"
ii) accept JSON {node_id:0,…} (values can be zero or 1)
iii) Recommend that next node based on configuration setting.
iv) If entered JSON meets 'success_criteria' , show a message "successful" along with next recommendation.
v) If entered JSON has all nodes set to "1", show a message "You aced it" and exit the program.
vi) have a hot "key word" to exit the recommender at any time.

Sequence of input is important for checking the performance of online recommender system. The question is presented in very ill-manner and lacks a lot of details. Some of them I have filled in using my own assumptions. Description of scenario at which it has to be used, and so whether a offline recommender system has to be created for Challenge 2 is unclear.

I will be happy to share description of the features used and other details of the algorithm if asked so.

References:

1. L. Breiman. Out-of-bag estimates. Technical report, 1996.
2. L. Breiman. Random forests. Machine Learning, 45(1):5– 32, October 2001.
3. On-line Random Forests - Amir Saffari Christian Leistner Jakob Santner Martin Godec Horst Bischof, Institute for Computer, Graphics and Vision, Graz University of Technology
4. Modeling User Behavior in Recommender Systems based on Maximum Entropy - Tomoharu Iwata, Kazumi Saito, Takeshi Yamada, NTT Communication Science Laboratories, Science City Kyoto 619-0237 Japan
5. On-line Random Naive Bayes for Tracking Martin Godec, Christian Leistner, Amir Saffari, Horst Bischof, Institute for Computer Vision and Graphics, Graz University of Technology, Graz, Austria

THE END