

# **Online Tutoring System for Learning Parsing Techniques**

*A thesis submitted*

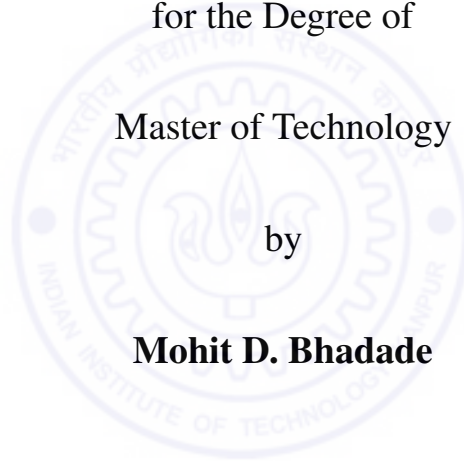
in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

**Mohit D. Bhadade**

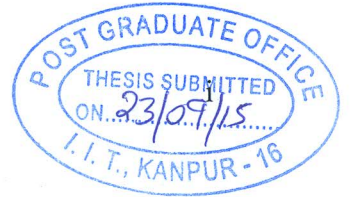


*to the*

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

September 2015



## CERTIFICATE

It is certified that the work contained in the thesis titled **Online Tutoring System for Learning Parsing Techniques**, by **Mohit D. Bhadade**, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

*Amey Karkare*

Prof Amey Karkare

Department of Computer Science & Engineering

IIT Kanpur

September 2015



## ABSTRACT

Name of student: **Mohit D. Bhadade**      Roll no: **13111033**

Degree for which submitted: **Master of Technology**

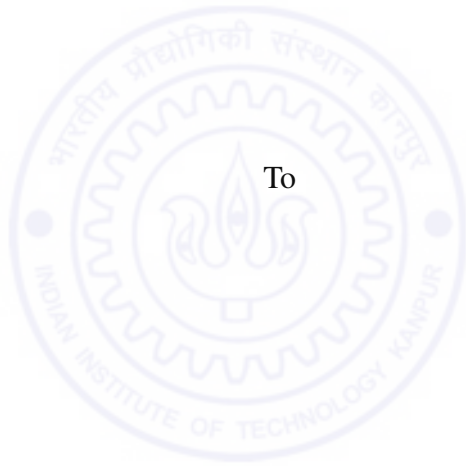
Department: **Computer Science & Engineering**

Thesis title: **Online Tutoring System for Learning Parsing Techniques**

Name of Thesis Supervisor: **Prof Amey Karkare**

Month and year of thesis submission: **September 2015**

In this thesis, we have developed an online interface for a tool for teaching parsing techniques for a compiler design course. The tool helps students interactively gain knowledge of various parsing techniques through multiple choice questions (MCQs) generated based on a Context Free Grammar (CFG) given as input. We use a back end console application and built a browser based front end for this system. Our system takes user interactivity into consideration and provides visually appealing platform to solve the problems related to parsing techniques. The interactive interface includes single page navigation, drag and drop interface, proper display of hints necessary to solve a particular question. We demonstrate our system on a website through a set of auto generated questions which are based on user's response. We believe that this browser-based interactive system can be used in online courses to allow users to learn at their own speed and from their own mistakes, with less dependence of human assistance.



# Acknowledgements

I would like to thank my thesis supervisor **Prof. Dr. Amey Karkare** for proposing the idea and directing me towards it. This project would have never been started had it not been for my thesis adviser Prof. Dr Amey Karkare's help and guidance. His support has been invaluable right from the starting phase. From his busy schedule, he would always find the time for the meeting and he would help me get out of the problem whenever I was stuck. My thesis would be incomplete without his help.

I am thankful to **Nimisha Agarwal** for her efforts in building core engine. Without her help it was impossible for me to integrate her system with my own.

I would like to thank **Adarsh Jagannatha** for giving his valuable input while building the base of the system and later helping me out with his expertise.

I am grateful to my **family** for being patient with me. They believed in my ability to work and gave their full support especially during final stretch of the work.

I appreciate the presence of all my **friends**, my **TP Buddies** group and batch mates here at IITK campus. I thoroughly enjoyed my workplace because of their presence.

# Contents

<b>List of Figures</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Statement . . . . .	2
1.2 Motivation . . . . .	2
1.3 Our Contribution . . . . .	3
<b>2 Background and Related Work</b>	<b>4</b>
<b>3 Technical Details</b>	<b>5</b>
3.1 Technical Architecture . . . . .	5
3.2 Working of Core Engine . . . . .	6
3.3 Data Transmission Model . . . . .	8
3.3.1 Need for Stateless to Stateful Converter . . . . .	8
3.3.2 Architecture of Data Transmission Model . . . . .	9
3.3.3 Using SPRING . . . . .	10
3.4 Web Interface and Development Strategies . . . . .	11
3.4.1 Web vs. Console . . . . .	11
3.4.2 Interfacing Techniques . . . . .	14
3.4.3 Development Strategies . . . . .	15
<b>4 User Manual</b>	<b>17</b>
4.1 Student as a User . . . . .	17
4.2 Admin as a User . . . . .	20

<b>5 Developer Manual</b>	<b>23</b>
5.1 Get access to the code . . . . .	23
5.2 Steps to Deploy . . . . .	23
<b>6 Conclusion And Future Work</b>	<b>25</b>
<b>References</b>	<b>26</b>



# List of Figures

1.1	Overall Idea . . . . .	2
3.1	Technical Architecture . . . . .	5
3.2	Different Modules in Core Engine . . . . .	8
3.3	Data Transmission Model . . . . .	9
3.4	Console Application . . . . .	12
3.5	Web Tutorial Home Page . . . . .	12
3.6	Web Application Front Page . . . . .	13
3.7	Web Application Interaction Page . . . . .	14
3.8	Drag And Drop Interface . . . . .	15
3.9	Drag And Drop Interface for Input Parsing String . . . . .	15
4.1	Quiz Links . . . . .	18
4.2	Warning Notifier . . . . .	19
4.3	Repeat Notifier . . . . .	20
4.4	Drag And Drop Interface . . . . .	20
4.5	Add/Remove User . . . . .	21
4.6	User History . . . . .	22



# Chapter 1

## Introduction

Advent of E-learning websites [1, 2] has been a boon to our education system. The hunger in human nature to learn more, and the recent technological advancement in Information Technology are the two most important factors responsible towards the development of online education. A decade back, personalized guidance of your teacher was considered as best possible way of learning. After a shift towards online education, onus of education lies on tutoring websites which therefore, tend to include more and more automation using Artificial Intelligence. We try and build expert systems to serve our motive. Systems which consists of all the domain information stored in it and the systems which are ready to accept more and more data, have obtained a significant importance over traditional approaches of learning [3]. And slowly the Intelligent Tutoring Systems (ITSs) became a centerpiece of automated online education.

Intelligent Tutoring Systems are available for variety of domains like Automata- DFA and NFA construction[4], Algebra[5], Geometry[6], and so on. According to researchers [7], ITSs consists of four basic components - **Domain model, Student model, Tutoring model, User Interface model**. In the later half of the thesis, we primarily focus on User Interface model. Designing an interface in an intuitive and aesthetic way is a part of user interface model.

## 1.1 Problem Statement

Given a console based application for teaching parsing techniques, problem is develop a front end for the same to put it to a proper use. Taking user interactivity into consideration, we need to develop an interactive platform. This thesis aims to integrate a back end machine (a console application for teaching parsing techniques) with the web and then display the content in a way which is easy to understand and answer. The thesis also aims to improve user interactivity and provide a web-based visually appealing platform. Figure 1.1 shows an overall purpose of the entire website. Our problem lies in the display part of the questionnaire.

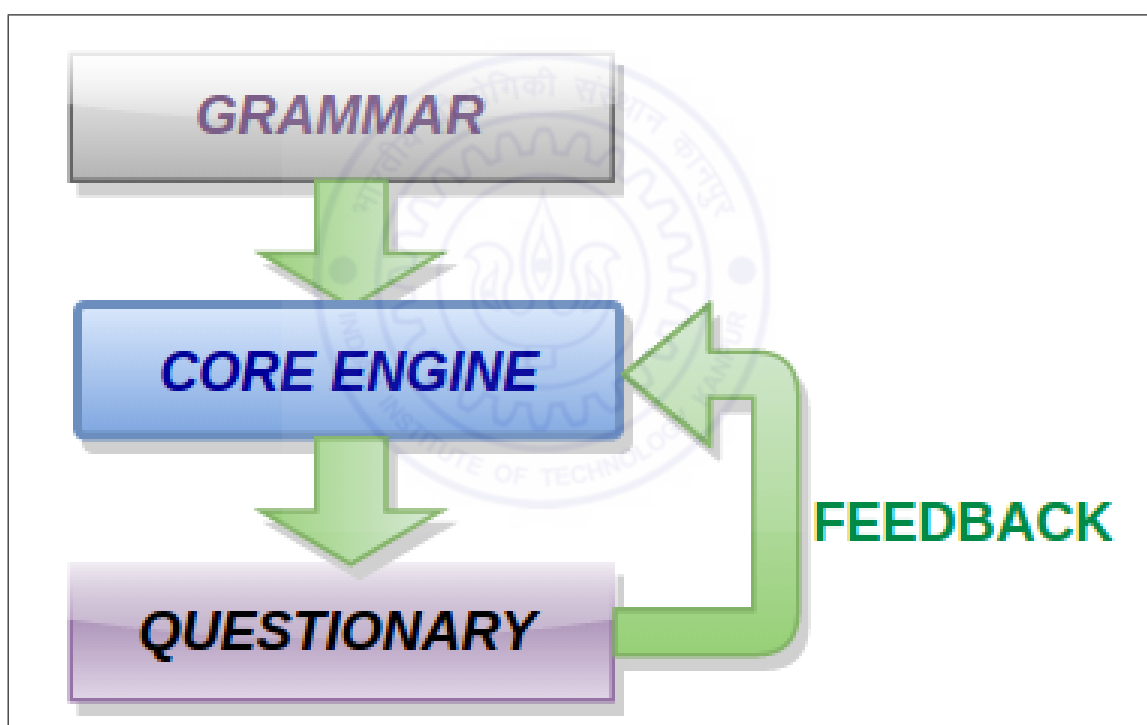


Figure 1.1: Overall Idea

## 1.2 Motivation

Recently, online education has become a key factor in student's learning. It is very important to create web based platforms for an easy use of any application which contributes towards teaching process. Providing a standard approach to connect such applications to user interactive platforms will help students learn faster in comparison to offline console

applications. Interactive platforms help students understand the concept properly.

While going through Autotutor[4] for DFA/NFA construction, we realized the significance of interface while solving a particular problem. The website provides a proper interface to draw automata using various symbols required for the same. This helps an user to concentrate more on the problem and worry less about the details of drawing it on the paper.

We had come across similar problems while learning parsing techniques. Complications involved in different parsing tables makes it difficult for a student to solve it online. Moreover, every problem requires some hints from a previously solved concept. To be specific, solving a LL parsing problem would require knowledge of first and follow tutorial. Displaying all the required hints together on a single page would surely help student focus more on the problem rather than worrying about other details. Building a front end with all these capabilities would surely serve the motive and help students learn faster.

### 1.3 Our Contribution

Our contribution involves three parts.

1. **Core Engine:** This part deals with compiler related problems which involves generation of hint questions, input parsing string, etc. The details will be seen in 3.2.
2. **Data transmission model:** This part mainly deals with interaction of stateful core engine with a completely stateless web interface. The model is the heart of the project. Refer 3.3 for details.
3. **Web Interface:** Our major contribution lies in this part. We deal with usability and the ease with which user can read and understand the problem. It also aims to provide self explanatory and aesthetically good user interface. Single page navigation including all the required set of hints, drag and drop interface, etc. are major contributions towards the project. Refer 3.4.2 for details.

# Chapter 2

## Background and Related Work

A possibility of existence of Intelligent Tutoring System have been discussed for centuries. Towards the end of second half of 20<sup>th</sup> century, research in Artificial Intelligence boosted the concept of intelligent tutoring and expert systems. A concept where a system could be trained and fed to work as a tutor would reduce the burden on available physical resources.

The biggest change in development of ITS took place with the advent of LISPITS[8]. Most dominant of various ITSs built over the period include Autotutor[9], Cognitive Tutor[10], etc.

Automata Tutor is a tool[11, 4] or rather a tutoring system which asks students to submit DFA/NFA as per the required statement. The animation involved for creating DFA/NFA attracts various students to solve problems on this portal. Adding to that, it also does an automatic grading of the submitted DFA/NFA. The feedback is provided in terms of grading which involves calculations of correct/incorrect states and various other factors related to DFA/NFA. Another such example is ESC101-ITS [12] developed at Indian Institute of Technology, Kanpur.

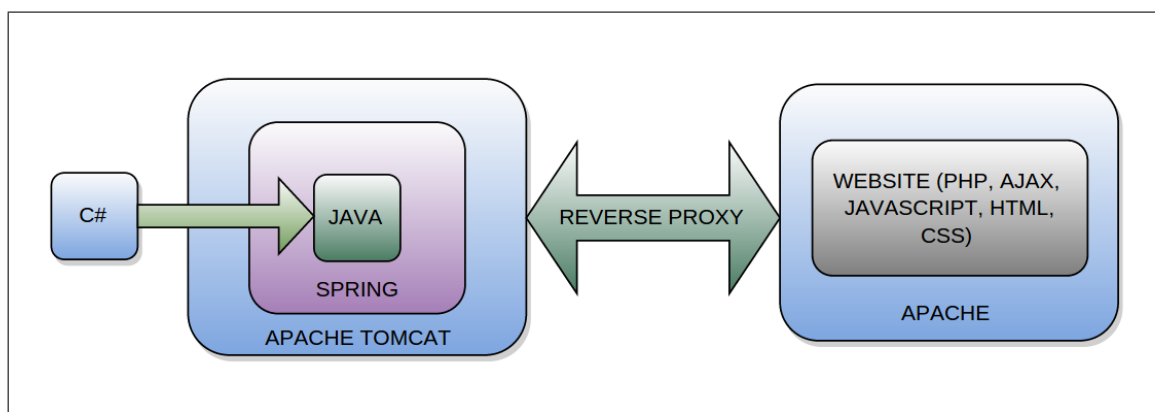
# Chapter 3

## Technical Details

### 3.1 Technical Architecture

In this section, we describe an overall flow and architecture of this thesis.

Initial code base for compiler specific calculations was written in C#. We felt the necessity of converting this code to Java as many frameworks were available which could link Java to web. At the front end, we built a website using PHP on top of Apache. Java's framework would work only on the top of Tomcat and so we needed an approach which would let us switch from Apache to Apache Tomcat and the other way round. Reverse proxy is one such method used to switch between Apache and Apache Tomcat. Figure 3.1 describes the technical architecture in brief.



**Figure 3.1:** Technical Architecture

Below is the code snippet for reverse proxy. Here, using **ProxyPass** and **ProxyPass-Reverse**, the requests for generating questions are being transferred between Apache and

Apache Tomcat.

```
ProxyRequests Off
ProxyPass /question http://localhost:8080/question
ProxyPassReverse /question http://localhost:8080/question

<Location "/question">
  # Configurations specific to this location. Add what you
  # need.
  # For instance , you can add mod_proxy_html directives to
  # fix
  # links in the HTML code.

  # Allow access to this proxied URL location for everyone.
  Order allow ,deny
  Allow from all
</Location >
```

Once the proxy bypass is ensured, the SPRING framework takes over for the task of data transmission from front end to back end and the other way round.

## 3.2 Working of Core Engine

Core engine [13] forms a major part of back end of the tutoring system. All the compiler related calculations are performed in this component of the system. Initially, entire core engine was built using C# but, later it was switched to Java with same features and set of calculations.

Core Engine takes user submitted Grammar as its primary input. Initial step in core engine involves grammar pre-processing. Grammar pre-processing involves the check on validity of grammar. Here, we expect the input grammar to be of the form in which non terminals are upper case letters and terminals are lower case. Each rule is identified using an arrow. Special symbol **epsilon** is used as it is while checking the grammar validity. Pre-processing involves calculations of various set of information required to solve entire questionnaire related to parsing techniques. It calculates set of terminals, non terminals, first set, follow set and canonical set.

Once these initial calculations are made, engine generates the first question based on

user's choice. For instance, if user wants to have a quiz on concept of "FIRST", then the engine generates the question for this category using a starting "non-terminal" from the set of non terminals generated prior to this. While generating the question, it also calculates the options that are to be displayed to user. It calculates a set of correct and incorrect answers for that particular question.

Most important calculations take place when user submits an answer to the question. Engine already maintains a state and therefore, it has prior knowledge on what question the answer corresponds to. Apart from this, it also knows actual correct and incorrect answers which are stored in one of the generated data structures during pre-processing. Core engine generates hint questions on the basis of a formula in which intersection of correct and incorrect set is taken with user's answer set. If the intersection of incorrect set with the answer set is empty and diff of correct set and answer set is empty(they are same), the result is that the answer is correct. In any other case, it finds out the missing/extra option submitted by user and starts providing hint questions for the same. Whenever user goes in a wrong direction, these questions lead him towards the correct answer.

Figure 3.2 shows how entire code is made modular and maintainable, and how different modules are responsible for calculations with respect to different concepts.

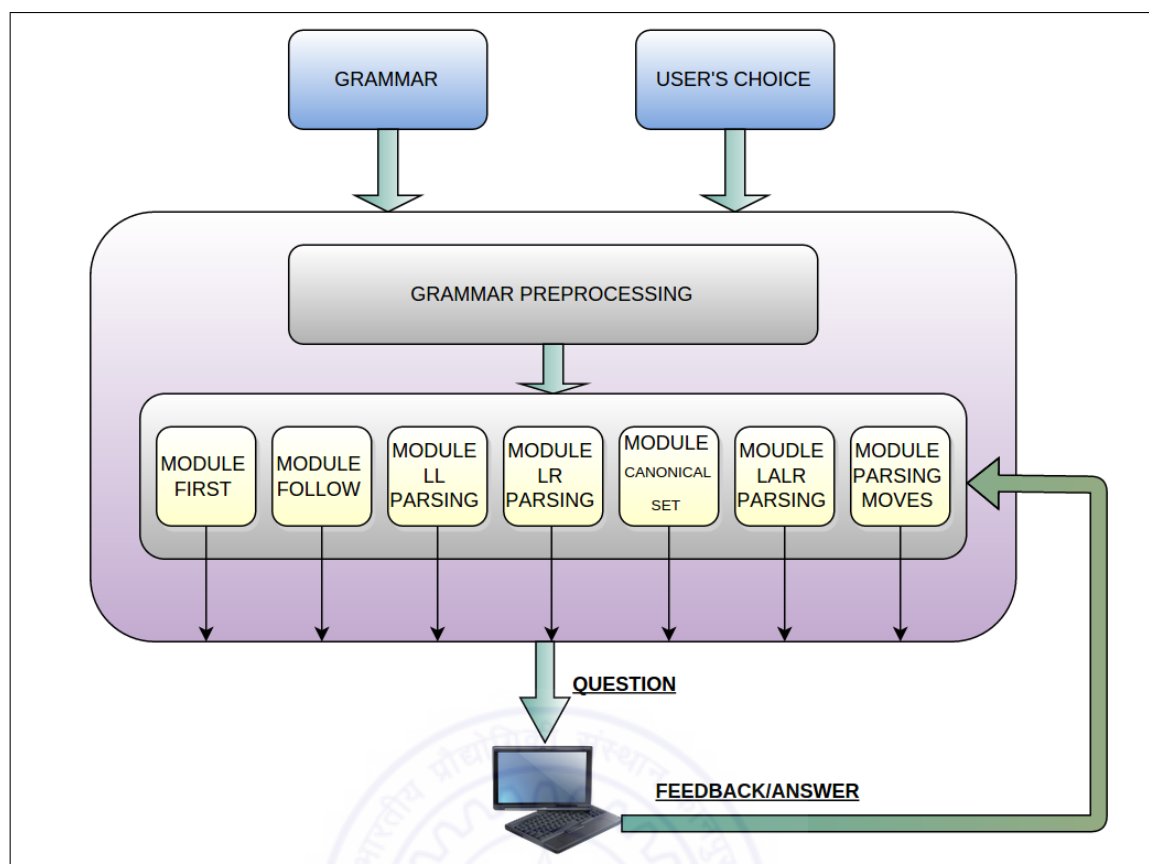


Figure 3.2: Different Modules in Core Engine

### 3.3 Data Transmission Model

We saw in previous section 3.2 how the calculations are made at the back end to generate the questions for quiz. It is very essential to have a model which will help us transmit this data to the front end where user shall view it on interactive platforms. All the data which we need to transmit from the back end are stored as Java objects. We require to build a model which will help us convert this data into a form readable to web. To solve all these issues, we build a data transmission model which is used to connect back end and front end.

#### 3.3.1 Need for Stateless to Stateful Converter

The existing core engine was previously built as a stateful machine. It means that core engine involved recursive calculations which would indirectly maintain a state for you where attributes inside a state could be considered as parameters to these recursive functions.



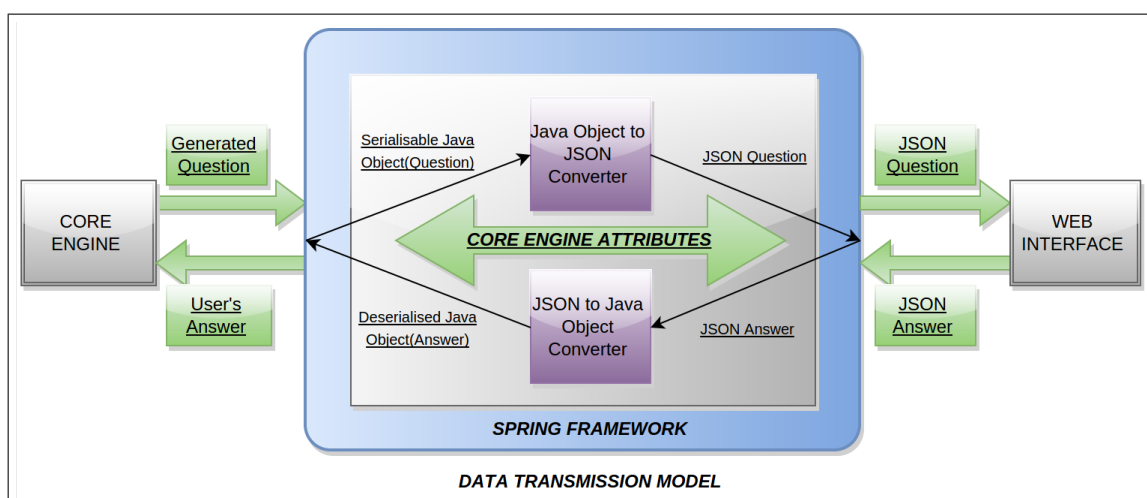
For example, consider a case in which a question is already generated(Q1 say) and we are supposed to generate Q2. Entire values generated out of pre-processing of grammar, answer to the question Q1, state of stack including all the terminals and non-terminals,etc. are considered to be a part of a single state.

Since HTTP is a stateless protocol, HTTP will forget all the state associated with it. Entire model of recursive calls has to be changed into a blackbox which takes user's choice(for a particular concept in the quiz), user's answer (except in first instance because question is yet to be generated) and grammar as an input and generates a question as an output. A middle ground was needed to store the entire state information.

Instead of storing this data using a particular medium, model could be built such that key attributes are transferred both from core engine to web interface and the other way round. This can virtually maintain a state even when you provide something as a web service. We present a data transmission model which helps us achieve the data transfer.

### 3.3.2 Architecture of Data Transmission Model

Data transmission model plays a key role in entire architectural and implementation setup of the project. Entire model helps us connect the stateful core engine with stateless web HTTP. Figure 3.3 is the architectural representation of the working of data transmission model.



**Figure 3.3:** Data Transmission Model

Data transmission model primarily involves the connection of web interface with the

back end by use of a **reverse proxy** protocol. Using this model, we fetch entire data in the JSON(Javascript Object Notation) format and convert it to Java readable and deserialized object. Once this answer is submitted to the core engine, data transmission model expects a reply back from the engine. The reply involves next question along with the parameters required to maintain the existing state. Various attributes/data structures like hash table, stack content, etc are being sent along with the question content. This provides a stateful behaviour to our web application.

### 3.3.3 Using SPRING

Spring framework is the heart of the data transmission model. As an initial step, we build a restful web service with spring. This service accepts HTTP GET requests at a particular URL. We consider this URL as a virtual URL as there is no physical page corresponding to it. To model a particular representation, we create a resource representation class. Then we create a resource controller as seen in the code snippet below.

```
@RestController
public class QuestionGenerator {
    @RequestMapping("/question")
    public QuestionFormat question(@RequestBody
        AnswerFormat .. {
```

The snippet above defines a controller class which is marks an entry point to the Java code base. Spring uses **RequestMapping** to map a particular implementation of a function to the virtual URL. So the main advantage is that the same function can be used to communicate to your web application using “QuestionFormat” and “AnswerFormat” as your Java to JSON convertible objects. Finally we build a executable JAR and run it using “gradle bootRun”

## 3.4 Web Interface and Development Strategies

### 3.4.1 Web vs. Console

Choosing a suitable option from a web application or a console application depends upon following parameters.

- Who is the target audience for the application?

Most non-technical users are unfamiliar with console or command line interfaces. Console apps are usually work well for technical users, since they are used to legacy console applications. Although we target a bunch of students who have ample knowledge of technicalities involved in using console interface, the primary aim is to make them learn in intuitive way. It is clear that graphical user interface(GUI) is a better approach to proceed with.

- What are we trying to achieve?

A standalone application with GUI may not serve well when it comes to keeping track of student's record. If we host the entire quiz on-line, it will help us keep track of the student.

As opposed to Web interfaces, console applications do not bring user interactivity into picture. You can test the working of your product on the console, but a naive user might find it difficult to interact with console for primary usage.

Adding to this, application becomes more scalable and usable. It becomes easier to keep track of things involved in your application.

- Images below gives a clear idea on how application looks when

1. viewed as a console application

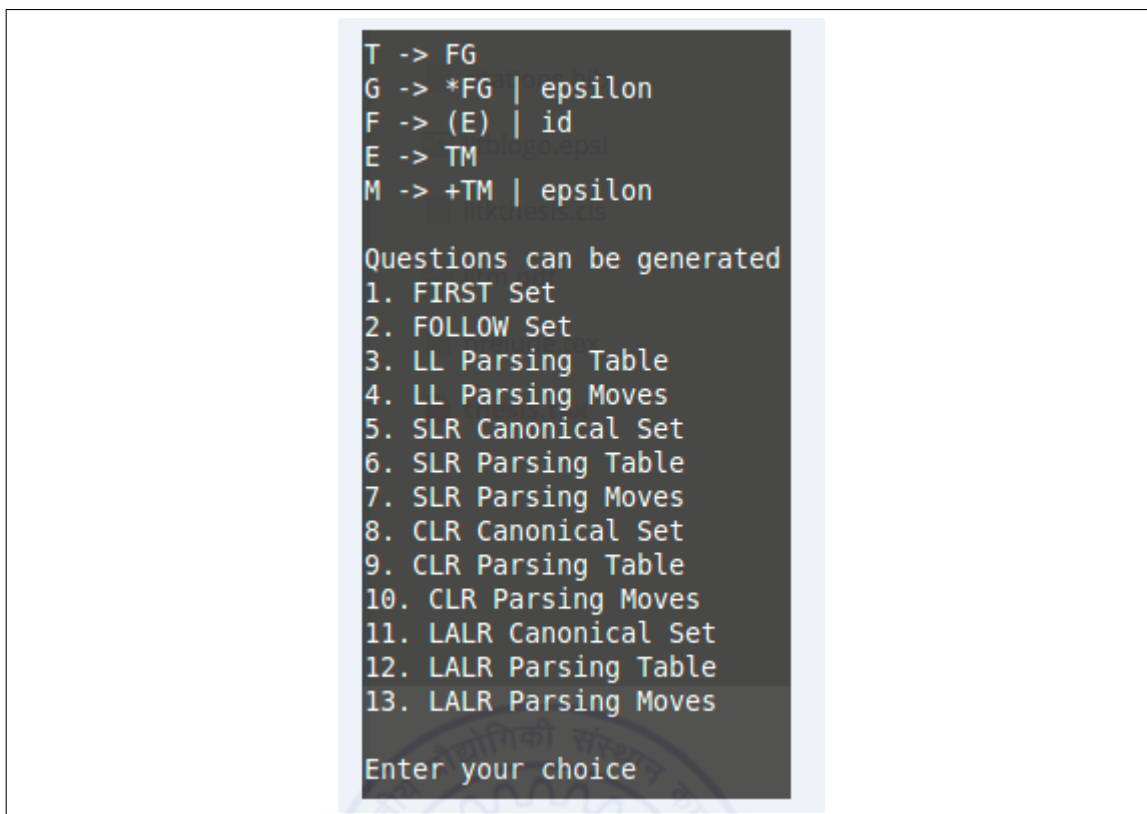


Figure 3.4: Console Application

## 2. viewed as a web application

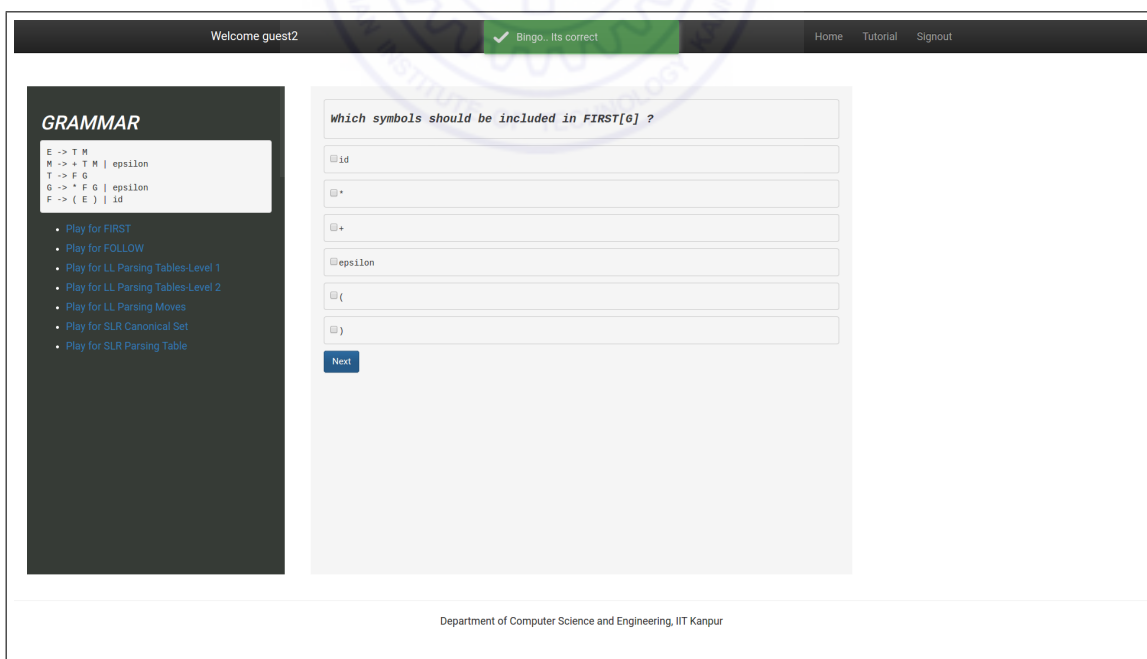


Figure 3.5: Web Tutorial Home Page

Welcome guest2 Home Tutorial Signout

## CSXXX : COMPILER TUTORIALS

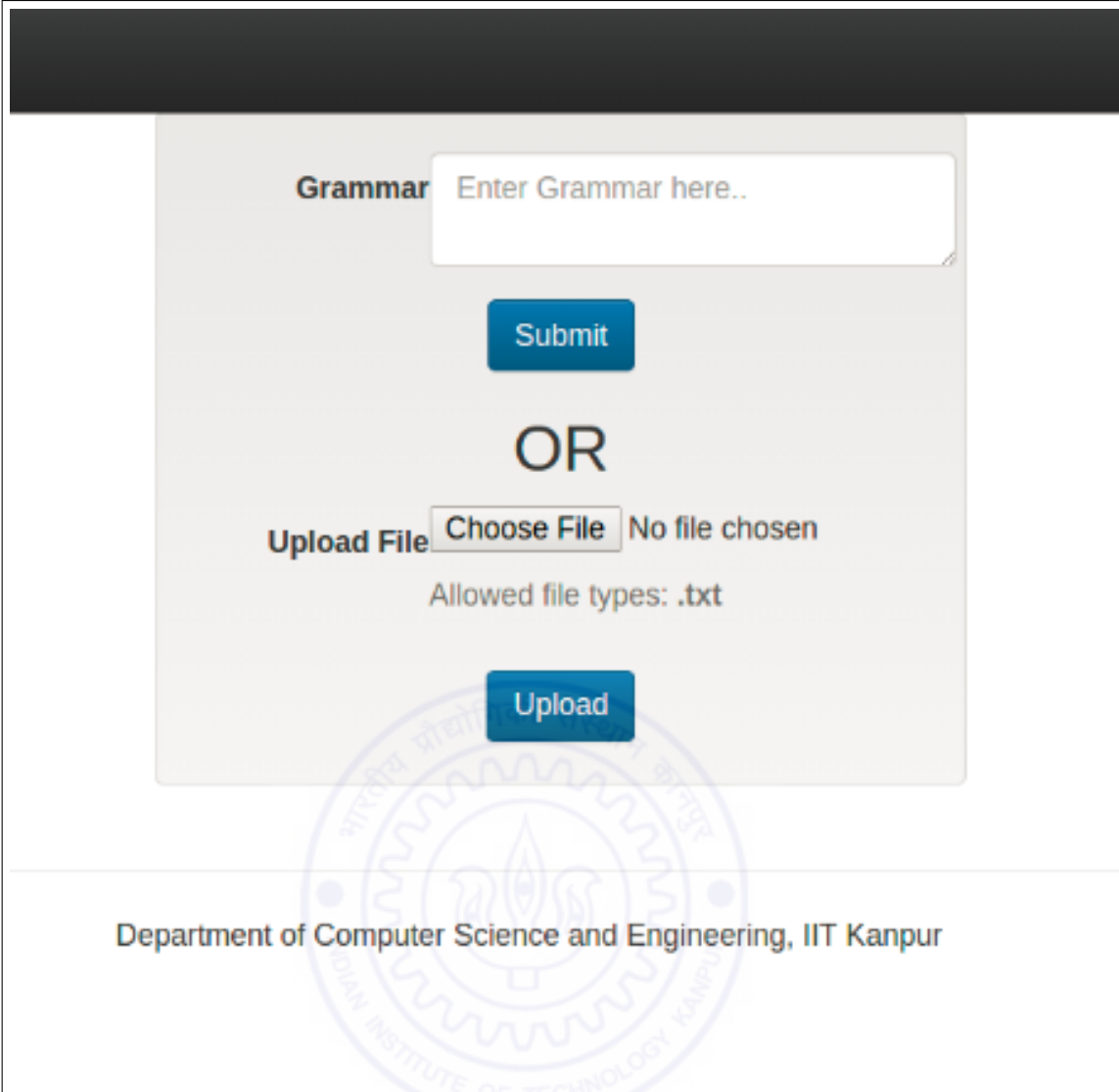
Instructor	Lecture Schedule	Course TAs
<b>Prof. Amey Karkare</b> Room RM karkare@cse.iitk.ac.in	Room KD101	<b>TA 1</b> ta1@cse.iitk.ac.in
	<b>Site Authors</b>	<b>TA 2</b> ta2@cse.iitk.ac.in
	<b>Mohit Bhadade</b> mohitdb@cse.iitk.ac.in	
	<b>Nimisha Agrawal</b> nimisha@cse.iitk.ac.in	

Department of Computer Science and Engineering, IIT Kanpur

172.27.30.43/pages#

**Figure 3.6:** Web Application Front Page





The image shows a web application interface with a dark header bar at the top. Below it, a light gray box contains the main interaction area. On the left, the word "Grammar" is displayed. To its right is a text input field with the placeholder text "Enter Grammar here..". Below the input field is a blue "Submit" button. In the center, the word "OR" is displayed in a large font. Below "OR", the text "Upload File" is followed by a "Choose File" button and the text "No file chosen". Below this is the text "Allowed file types: .txt". At the bottom of the light gray box is a blue "Upload" button. A large, faint watermark of the IIT Kanpur logo is visible in the background. At the bottom of the page, the text "Department of Computer Science and Engineering, IIT Kanpur" is displayed.

Figure 3.7: Web Application Interaction Page

### 3.4.2 Interfacing Techniques

This thesis aims to provide a single page navigation platform where user can find all the required hints to solve a particular problem. While studying the concepts involving LL table, parsing input string, SLR table, etc. students usually need a platform where they could solve problems with a “proper user interaction”. Here, we come up with a **Drag and Drop** interface as a solution to the problem. The solution is also inspired by Autotutor[9] which allows such innovative interface for constructing DFA/NFA. Figure 3.8 and Figure 3.9 gives a snapshot of how this interface looks.

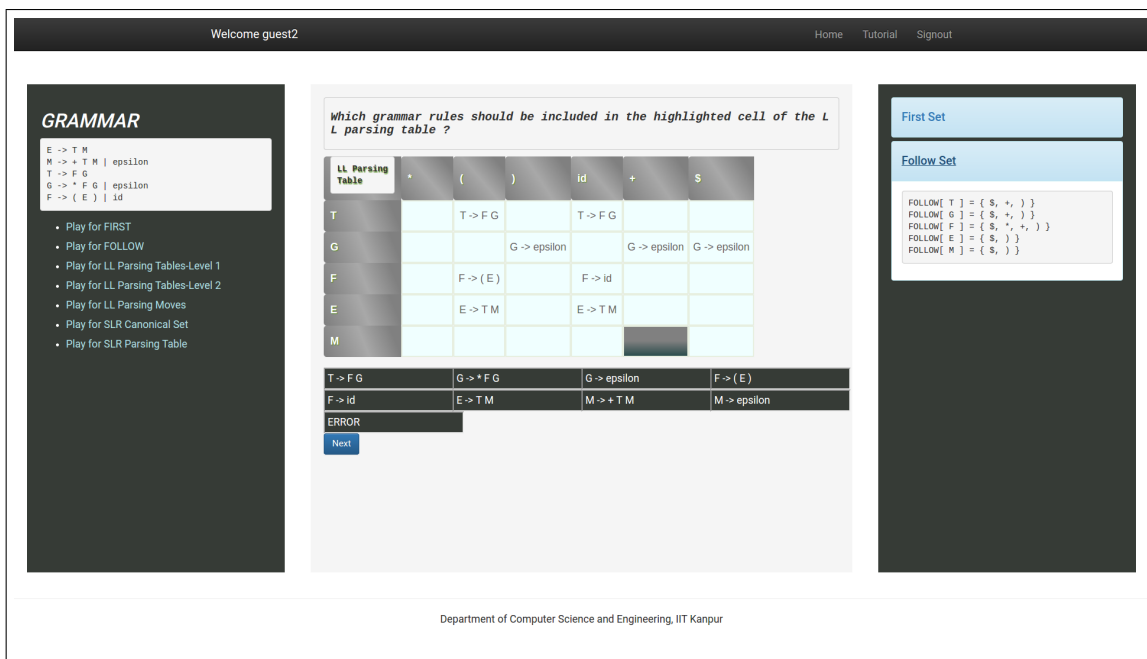


Figure 3.8: Drag And Drop Interface

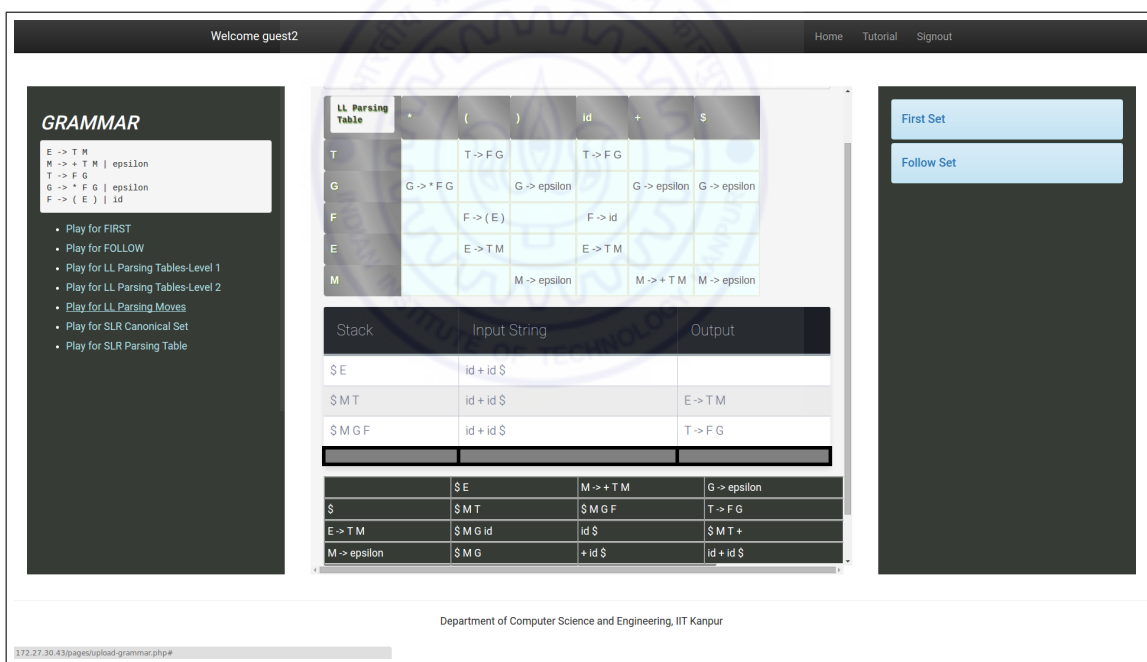


Figure 3.9: Drag And Drop Interface for Input Parsing String

### 3.4.3 Development Strategies

#### Code Modifiability

- While we develop the entire project primary concern lies in the issue of modifiability. It means the ease with which one can modify the code for additional functionality.

- We achieve this through data transmission model. The idea is to transmit all the relevant data back and forth along with the request and response. So, whenever you need to modify something, you simply need to add a parameter to this protocol.
- The code snippet below shows the transfer object used by transmission model at the front end of the website.

```

window.getQuestion({ 'questionReturned':'', 'useranswer':[], 'nonterminals':[], 'incorrect':[], 'correct':[], 'wrong':'', 'right':'', 'state':'', 'sym':'', 'uname':getUname(), 'ch': window.ch, 'l1cellstk':[], 'l1cell':[], 'cellnoset':[], 'tableanswer':[], 'tablerow':'', 'tablecol':''});

```

### **Crux of the Website**

- The file questionCall.js contains the most important client side scripting code. It is responsible for all the necessary functionality of the project.
- The quiz loads with an AJAX call to the virtual URL. This redirects the call to core engine with the use of reverse proxy setting.
- The code handles all the cases corresponding to different links of the quiz that are available to you.
- This file is backbone of logic behind all the created user interface. This includes scripting behind the drag and drop tables, input parsing tables, notifiers, etc.
- The easiest way to understand client side scripting code is to know the contents of this file. Please refer Developer Manual on how to get access to the code.



# Chapter 4

## User Manual

### 4.1 Student as a User

- The admin of the site registers students and provides credentials to them at the start of the course. Student can change the password at any stage later.
- Once you log in successfully, you will be redirected to the home page of the web site. The tutorial section at the upper right corner of the page should be used by the user to undertake the quiz.
- In the tutorial section, first thing you need to do is upload a Grammar. You can do this either by selecting a text file from a list of admin uploaded files or uploading your own text file.
- You will be redirected to the quiz page once you upload the grammar. Here, towards the left side of the page you will see the set of links which corresponds to different concepts involved in learning of a compiler.

Welcome guest2

## GRAMMAR

```

E -> T M
M -> + T M | epsilon
T -> F G
G -> * F G | epsilon
F -> ( E ) | id

```

- [Play for FIRST](#)
- [Play for FOLLOW](#)
- [Play for LL Parsing Tables-Level 1](#)
- [Play for LL Parsing Tables-Level 2](#)
- [Play for LL Parsing Moves](#)
- [Play for SLR Canonical Set](#)
- [Play for SLR Parsing Table](#)

# Links for Quiz

Figure 4.1: Quiz Links

- Links **Play for First** and (Play for Follow) is for First and Follow Tutorial.
- When you click on a link **Play for LL Parsing Table-Level 1** or **Play for LL Parsing Table-Level 2**, you will come across a parsing table. Here, you are supposed to drag the option and drop it in the corresponding highlighted cell. You can drop multiple options in these cells. Do not click submit unless you have dropped all the

answer which you think are correct.

- Finally, whenever you submit an answer corresponding to any question, you will be notified if your answer is correct. When you are going in wrong direction, it will ask you to reconsider your choice and give you a warning notifier.

The screenshot shows a web application interface with a dark header. The header contains the text "Welcome guest2" on the left, a warning icon and the text "Whoa! Look Again" in the center, and navigation links "Home", "Tutorial", and "Signout" on the right. The main content area is divided into two sections. On the left, there is a sidebar titled "GRAMMAR" with a list of grammar rules: E → T M, N → + T M | epsilon, T → F G, G → \* F G | epsilon, and F → ( E ) | id. Below the rules is a list of links: "Play for FIRST", "Play for FOLLOW", "Play for LL Parsing Tables-Level 1", "Play for LL Parsing Tables-Level 2", "Play for LL Parsing Moves", "Play for SLR Canonical Set", and "Play for SLR Parsing Table". The main content area displays a question: "According to which of the following rules, '(' is a part of FIRST[G] ?". Below the question are four numbered options: 1. If X is a terminal, then FIRST(X) is {X}. 2. If X is nonterminal and X → alpha is a production, then add alpha to FIRST(X). If X → epsilon is a production, then add epsilon to FIRST(X). 3. If X → Y1Y2...Yk is a production, then for all i such that all of Y1, ..., Yi-1 are nonterminals and FIRST(Yj) contains epsilon for j = 1, 2, ..., i-1 (i.e. Y1Y2...Yi-1 → epsilon), add every non-epsilon symbol in FIRST(Yi) to FIRST(X). If epsilon is in FIRST(Yj) for all j = 1, 2, ..., k, then add epsilon to FIRST(X). 4. No valid rule for this symbol. Below the options are radio buttons for each option and a "Next" button. A large, faint watermark of the Indian Institute of Technology Kanpur logo is visible in the background of the main content area. At the bottom of the page, the text "Department of Computer Science and Engineering, IIT Kanpur" is displayed.

**Figure 4.2:** Warning Notifier

Upon giving a wrong answer, you might be asked about the same question again and this will be notified to you. All the notifications mentioned above will appear at the top centre of the web page.

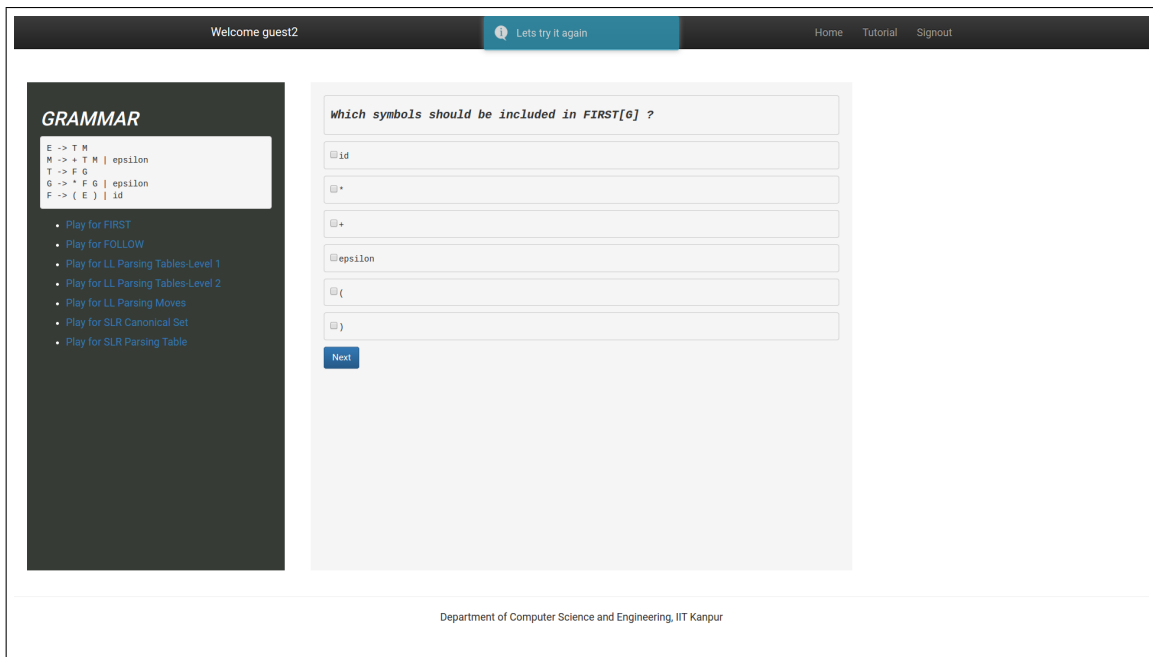


Figure 4.3: Repeat Notifier

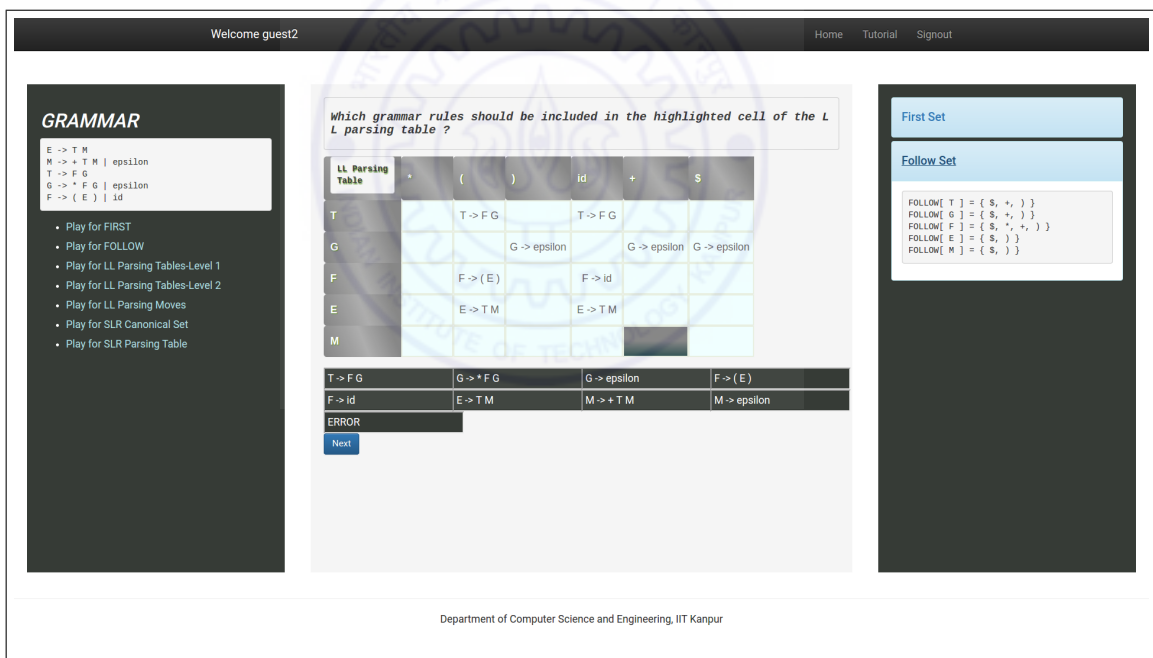


Figure 4.4: Drag And Drop Interface

## 4.2 Admin as a User

- Admin can add/remove any user according to his choice.

The screenshot shows a web interface for user management. It has a dark header bar at the top. Below it, there is a light gray panel containing the following elements:

- Username** label next to an input field containing the text "Username".
- Password** label next to an input field containing the text "Password".
- A blue button labeled **Add Student**.
- Remove User** label next to an input field containing the text "Enter username to be removed".
- A blue button labeled **Remove Student**.

Below the gray panel, there is a white section with the text "Department of Computer Science and Engineering, IIT Kanpur". A large, faint watermark of the IIT Kanpur logo is visible in the background of this section.

**Figure 4.5:** Add/Remove User

- Admin can look at the history of all the registered users. Admin can know what all sets of grammars are being uploaded and practised by a particular user.

Select User ▾

### Grammar records for guest

- guestgrammar4.txt

```
S -> L = R | R
L -> * R | id
R -> L
```
- guestgrammar6.txt

```
E -> T + E | T
T -> 0 | 1
```
- guestgrammar1.txt

```
E -> T M
M -> + T M | epsilon
T -> F G
G -> * F G | epsilon
F -> ( E ) | id
```

Department of Computer Science and Engineering, IIT Kanpur

**Figure 4.6:** User History

- Admin can upload his own grammar and inform students to practise the questions based on these grammars. All the users can either upload their own grammar or select a grammar from a list of grammars uploaded by admin.

# Chapter 5

## Developer Manual

### 5.1 Get access to the code

- You can get the access to entire website code at <https://github.com/mohitbhadade/Intelligent-Tutoring>.
- Branch “server” contains website code.
- Do a “git clone <clone link>” on your machine to get the code.

### 5.2 Steps to Deploy

- Once you have the code, first step is to install Apache and Apache Tomcat on your machine. Installation procedures may vary slightly according to the underlying operating system.
- You need to add a config file to Apache configuration. This file handles all the reverse proxy settings mentioned in 3.1
- After this initial setup, you need to install **SPRING** and build a restful service. The spring guide[14] will help you understand the parts of the code which involve use of important keywords related to spring.

- After this, you can put your entire Java code(this part is like some java utility which you need to connect to a website) in “gs-rest-service/initial/src”. Perform a bootRun using “./gradlew bootRun” to get the instance of core engine running. Your code in questionCall.js will vary as per your Java files and their respective function parameters. We use a core engine which contains a dynamic questionnaire for compiler concepts.
- You shall be ready with the setup on your machine. Run localhost on your browser and website shall be up.





# Chapter 6

## Conclusion And Future Work

We implemented an Intelligent Tutoring System which takes a grammar as an input and generates a quiz dynamically on particular concept of the compiler. In addition to this, the questions in the quiz are also generated based upon the user's answer. This feedback based ITS will surely help students grasp the concepts of compiler at a speed greater than usual.

We can add a suitable interface to display a graph representing canonical set for SLR parsing. The current problem lies in the fact that the graph can have cross edges and it will obtrude when we try to display it. We need to find a **near to planar** graph to display it for proper use. Addition of other innovative and appealing interfacing techniques will depend upon feedback of the user. Also, current transmission model aims at developing a web based application. This can be generalized by providing a set of API's for an application based on any platform.

# References

- [1] *Coursera*. URL: <https://www.coursera.org/>.
- [2] *Khan Academy*. URL: <https://www.khanacademy.org/>.
- [3] *ITS Significance*. URL: [http://edutechwiki.unige.ch/en/Intelligent\\_tutoring\\_system](http://edutechwiki.unige.ch/en/Intelligent_tutoring_system).
- [4] Rajeiv Alur, Loris D'Antoni, Sumit Gulwani, Dileep Kini, and Mahesh Viswanathan. "Automated Grading of DFA Constructions". In: *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. Ed. by Francesca Rossi. IJCAI/AAAI, 2013. ISBN: 978-1-57735-633-2. URL: <http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6759>.
- [5] Rohit Singh, Sumit Gulwani, and Sriram K. Rajamani. "Automatically Generating Algebra Problems". In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. 2012. URL: <http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5133>.
- [6] Chris Alvin, Sumit Gulwani, Rupak Majumdar, and Supratik Mukhopadhyay. "Synthesis of Geometry Proof Problems". In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*. 2014, pp. 245–252. URL: <http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8617>.
- [7] *Wikipedia ITS*. URL: [http://en.wikipedia.org/wiki/Intelligent\\_tutoring\\_system](http://en.wikipedia.org/wiki/Intelligent_tutoring_system).
- [8] *LISPITS*. URL: [http://act-r.psy.cmu.edu/wordpress/wp-content/uploads/2012/12/167corbett\\_and\\_anderson\\_lisp.pdf](http://act-r.psy.cmu.edu/wordpress/wp-content/uploads/2012/12/167corbett_and_anderson_lisp.pdf).
- [9] *AutoTutor*. URL: <http://en.wikipedia.org/wiki/AutoTutor>.
- [10] *Cognitive Tutor*. URL: [http://en.wikipedia.org/wiki/Cognitive\\_tutor](http://en.wikipedia.org/wiki/Cognitive_tutor).
- [11] *Automata Tutor*. URL: <http://www.automatatutor.com/about/>.
- [12] Rajdeep Das. *A Platform for Data Analysis and Tutoring for Introductory Programming*. 2014-2015. URL: <http://172.28.64.70:8080/jspui/handle/123456789/14911>.
- [13] Nimisha Agrawal. *A Tool for Teaching Parsing Techniques*. 2014-2015. URL: <http://172.28.64.70:8080/jspui/handle/123456789/15270>.
- [14] *Spring Guide*. URL: <http://spring.io/guides/gs/rest-service/>.